FASAC

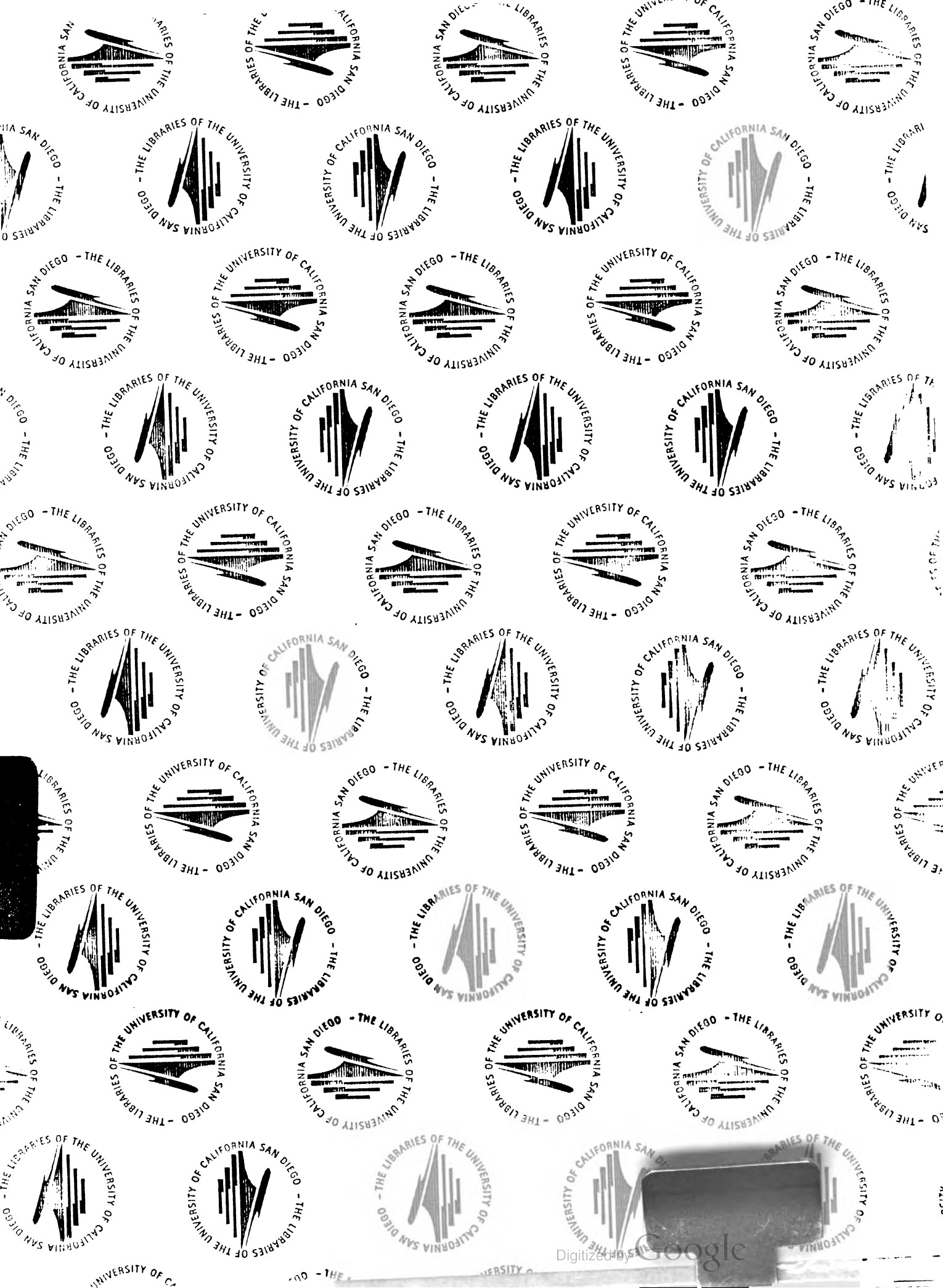
FOREIGN APPLIED SCIENCES ASSESSMENT CENTER
TECHNICAL ASSESSMENT REPORT

PARALLEL PROCESSING RESEARCH IN THE FORMER SOVIET UNION

J. J. Dongarra
L. Synder
P. Wolcott

March 1992

SAIC® An Employee-Owned Company
Science Applications International Corporation


THE LIBRARIES OF THE UNIVERSITY OF CALIFORNIA SAN DIEGO
THE UNIVERSITY OF CALIFORNIA SAN DIEGO - THE LIBRARIES
Digitized by Google

FASAC Technical Assessment Report

PARALLEL PROCESSING RESEARCH IN THE FORMER SOVIET UNION

J. J. Dongarra
L. Snyder
P. Wolcott

March 1992

Science Applications International Corporation
1710 Goodridge Drive, P. O. Box 1303, McLean, VA 22102
10260 Campus Point Drive, San Diego, CA 92121

(blank)

ABSTRACT

This technical assessment report examines strengths and weaknesses of parallel processing research and development in the Soviet Union during the 1980s (through the first half of 1991). The assessment was carried out by a panel of US scientists who are experts on parallel processing hardware, software, algorithms, and applications, and on Soviet computing.

Soviet computer research and development organizations have pursued many of the major avenues of inquiry related to parallel processing that their colleagues in the West have chosen to explore. However, the limited size and substantial breadth of their effort have limited the collective depth of Soviet activity. Even more serious limitations (and delays) of Soviet achievement in parallel processing research can be traced to shortcomings of the Soviet computer industry, which was unable to supply adequate, reliable computer components to the researchers. Without the ability to build, demonstrate, and test embodiments of their ideas in actual high-performance parallel hardware, both the scope of activity and the success of Soviet parallel processing researchers were severely limited.

The quality of the Soviet parallel processing research assessed varied from very sound and interesting to pedestrian, with most of the groups at the major hardware and software centers to which the work is largely confined doing good (or at least serious) research. In a few instances, interesting and competent parallel language development work was found at institutions not associated with hardware development efforts. Unlike Soviet mainframe and minicomputer developers, Soviet parallel processing researchers have not concentrated their efforts on reverse-engineering specific Western systems. No evidence was found of successful Soviet attempts to use breakthroughs in parallel processing technology to "leapfrog" impediments and limitations that Soviet industrial weakness in microelectronics and other computer manufacturing areas impose on the performance of high-end Soviet computers.

At the beginning of the 1990s, powerful Western processors began to become available for rubles within the Soviet Union. Use of these processors should allow developers of parallel processing hardware in the successor states of the former Soviet Union to evade the shortcomings of indigenous microelectronics and manufacturing capabilities, shortening development cycles and placing reliable systems into the hands of software developers and users quickly. The availability of Western components is also allowing new groups to become involved in parallel computer development. Because Soviet developers of parallel programming languages, systems, and applications were able to advance surprisingly far with limited access to parallel hardware, if Western or Japanese computers become available to fill this void, well-trained researchers in the successor states of the former Soviet Union should be ready to put them to effective use.

(blank)

PARALLEL PROCESSING RESEARCH IN THE FORMER SOVIET UNION

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

(blank)

FOREWORD

This report, *Parallel Processing Research in the Former Soviet Union,* is one in a series of technical assessment reports produced by the Foreign Applied Sciences Assessment Center (FASAC), operated for the Federal government by Science Applications International Corporation (SAIC). These reports assess selected fields of foreign basic and applied research, evaluate and compare the state of the art in the country or area of interest with US and world standards, and identify important trends that could lead to future applications of military, economic, or political importance. This report, like others produced by the Center, is intended to enhance US knowledge of foreign applied science activities and trends, to help reduce the risk of technology transfer, and to provide a background for US research and development decisions. Appendix E lists the FASAC reports completed and in production.

This report was prepared by a panel of nationally recognized scientists who are active in computer science research:

• Dr. Jack J. Dongarra	Distinguished Scientist Department of Computer Science University of Tennessee, and Mathematical Sciences Section Oak Ridge National Laboratory
• Dr. Lawrence Snyder	Professor Department of Computer Science and Engineering University of Washington
• Peter Wolcott	PhD candidate in Management Information Systems University of Arizona.

The assistance of Dr. Seymour E. Goodman, Professor of Management Information Systems and Director of International Business Programs at the University of Arizona, who served as senior advisor to the panel, is gratefully acknowledged.

On a part-time basis over the period from April to December 1991, each panel member spent a substantial amount of time assessing published Soviet literature on

parallel processing research. Efforts were made by the panel and by SAIC to identify and acquire as many relevant Soviet publications as possible, subject to constraints of availability and search criteria. Principal Soviet technical publications reviewed by the panelists are listed in Appendix C.

EXECUTIVE SUMMARY

This technical assessment report examines strengths and weaknesses of parallel processing research and development in the Soviet Union during the 1980s (through the first half of 1991). This work may continue in the successor states that emerged at the end of 1991 if support for the work is continued by the new governments. It will be the foundation for any efforts to develop parallel processing technology the new nations may choose to pursue.

Parallel processing research aims to develop and improve computing hardware and software that use multiple processing units to increase the overall speed of demanding computations, by executing as many operations as possible simultaneously rather than sequentially. This strategy for acceleration of computation is being pursued in the hope that it will evade present physical and engineering speed limits for conventional sequential (von Neumann) computers. To some extent, that hope has already been realized in the fastest computers commercially available in the West, Japan, and (to a much more limited extent) the former Soviet Union.

Obstacles to use of parallelism as a strategy for accelerating computation are of two kinds: limits imposed by the fact that some parts of every problem must be solved in sequence, and limits imposed by the fact that preparation for parallel computation and communication between processes executing in parallel themselves consume some of the time parallelism is intended to save. Parallel processing research is concerned with identifying those problems for which the necessarily sequential portion is potentially small, and devising effective and efficient ways of extracting and carrying out the parts of the computation that can be accelerated by use of multiple processors. One of the key findings of many years of research and practice in the West is that achieving high performance via straightforward application of parallelism has not been easy, as some had naively hoped.

Soviet computer research and development organizations have pursued many of the major avenues of inquiry related to parallel processing that their colleagues in the West have chosen to explore. However, the limited size of this effort and its substantial breadth have combined to limit the collective depth of Soviet parallel processing research.

At the beginning of the development cycle, the ideas embodied in Soviet parallel processing hardware and software have often been at the leading edge of worldwide research. However, severely limited Soviet ability to build, demonstrate, and test embodiments of these ideas has delayed and limited Soviet achievement. Lifting or relaxation of past and existing restrictions on Soviet imports of powerful hardware could dramatically change this situation, if financial support for purchase of foreign machines, components, and software tools can be obtained. The desire and ability of the successor states of the former Soviet Union to support long-term research like parallel processing development have yet to be established.

The quality of the Soviet parallel processing research assessed varied from very sound and interesting to pedestrian, with most of the groups at the major hardware and software centers to which the work is largely confined doing good (or at least serious) research. In a few instances, interesting and competent parallel language development work was found at institutions not associated with hardware development efforts. Unlike Soviet mainframe and minicomputer developers, Soviet parallel processing researchers have not concentrated their efforts on reverse-engineering specific Western systems. No evidence was found of successful Soviet attempts to use breakthroughs in parallel processing technology to "leapfrog" impediments and limitations that Soviet industrial weakness in microelectronics and other computer manufacturing areas impose on the performance of high-end Soviet computers.

HARDWARE

Soviet development of mainframe, mini-, and micro-computers was characteristically imitative, striving to produce functional equivalents of specific Western computer series. By contrast, Soviet attempts to develop parallel processing machines have shown a considerable degree of originality. Only one high-performance Soviet computer, the vector-pipelined Elektronika-SSBIS (the "Red Cray"), can be considered a close analog of a Western machine.

The El'brus and PS-2x00 projects, carried out within industrial ministries, show the inspiration of Western developments. These were mid-term industrial efforts to produce multiprocessor machines to improve Soviet high-performance computing capabilities. High-level Western design ideas were generally followed, but in each

case some features of the implementation were indigenous and in several cases problems in Western designs were overcome. About 50 El'brus-2 machines were installed in the second half of the 1980s, with serial production scheduled to continue.[1] About 200 PS-2000 machines were installed in Soviet scientific and technical institutions in the 1980s. The PS-2100, a more powerful machine with semiconductor external storage, began shipping in the late 1980s. Because no Soviet floating point hardware units were available, the PS-2100's designers were forced to use programmable gate arrays for floating point computations.

There were half a dozen major projects to develop advanced multiprocessor computer systems in the Soviet Union in the 1980s. Several of these efforts began under the sponsorship of the research and development arm of the manufacturer of Unified System (ES) mainframes. These projects include research in most major categories of parallel architecture, including coarse-grain shared memory systems with relatively few processors, single instruction stream, multiple data stream (SIMD) machines, dataflow and/or reduction machines, and reconfigurable architectures. As in the overall Soviet parallel processing effort, the price of this breadth of hardware research was a program of limited depth, without multiple research groups pursuing similar ideas.

These groups have done their work on what would be considered unusually long development cycles in the West. Though many of their ideas were at the world state of the art when they were conceived, the resulting functioning Soviet parallel processing hardware lagged badly.

The inter-processor interconnect schemes represented in the products of these research efforts have shown limited variety. Only one example of a hypercube topology was seen—and this only recently—and no examples of butterfly interconnection. The one hypercube connection experiment may indicate the beginning of a serious effort to develop massively parallel (>1,000 processors) systems.

---

1 Note added in proof: At the end of February 1992, Sun Microsystems announced a contract with the El'brus advanced development team to support software development work of interest to Sun by about 50 members of the team in Moscow.

Soviet research on dataflow and reduction machines reflects some ideas and approaches not found in Western machines. Recent funding reductions and the general economic crisis in the successor states of the former Soviet Union make the current status of these efforts unclear.

All Soviet parallel processing hardware development has been restrained by an inadequate component base in Soviet industry. Some major Soviet parallel processing hardware projects have suffered from pushing too many Soviet technological boundaries at once. Shortfalls in individual areas then delayed the overall effort. In addition, systems built around a shaky advanced component base have been predictably unreliable. Architecture features designed to provide redundancy and fault-tolerance have been only partially successful in overcoming this weakness.

Relief from component reliability problems began to appear in the mid-1980s, when use of new, relatively reliable, relatively high-performance Soviet gate arrays led to significant improvements in performance and reliability of new Soviet multiprocessor computers. At the beginning of the 1990s, powerful Western processors such as the Intel 386, 486, and i860 microprocessors, the Motorola 68030 and 68040 microprocessors, and Inmos transputers began to become available for rubles within the Soviet Union. Use of these processors will allow developers of parallel processing hardware to evade the shortcomings of indigenous microelectronics and manufacturing capabilities, shortening development cycles and placing reliable systems into the hands of software developers and users quickly. The availability of Western components has also allowed new groups to become involved in parallel computer development.

European emphasis on transputers (processors manufactured by the British company Inmos that have built-in interprocessor communication links) seems to be producing a strong echo in parallel processing research in the successor states of the former Soviet Union. Nearly all of the major Soviet parallel processing groups, which had traditionally designed their processors using the technology base available in the Soviet Union, reportedly are seeking to implement some of their machine design and software ideas on transputer platforms.

Economic troubles in the successor states of the former Soviet Union have harmed all the major parallel processing hardware projects. Support for fundamental research in high-performance computing has remained relatively constant over the last few years, although the true value of this support has been eroded by accelerating inflation. Several hardware development projects have been terminated, and those that remain are having increasing problems getting support for research and manufacturing.

PROGRAMMING

Conventional computer programming specifies a sequence of steps that will solve a problem. For parallel computation, it is also necessary to determine how these steps can be divided up among independent processors and applied to the common data without "unexpected collisions." Exponential explosion of the number of potential interactions qualitatively changes programming, generally precluding pre-planning of computations by the programmer at the lowest level of detail. The complexity of programming, already substantial in the sequential case, increases dramatically. Parallel programming research invents and evaluates new languages, language constructs, and programming tools (including compilers) needed for expressing the new features and carrying out the new tasks.

Soviet research in parallel programming has addressed many of the same topics that are actively pursued in the West and Japan. Generally, the Soviet work has extended Western work rather than broken new ground. Soviet research groups have been active in automatic vectorization/parallelization (via translators that attempt to discover and exploit parallelism automatically); in parallel programming languages (which attempt to make the parallel programming task easier, use a parallel architecture effectively, or both); and (to a more limited extent) in development of parallel programming environments and tools.

In several instances, the work is of high quality, though none of this work is superior to comparable efforts in the West and Japan. Limited experience with actual hardware is evident in the content of the work, and experimental results are seldom described. Perhaps conversely, more explicit attention is devoted to mathematical foundations than is usual in the West or Japan. This analytic sophistication and the quality of some of the work (despite its limitations) suggest that several groups

could rapidly make good use of reliable parallel hardware, were it to be made available to them.

As in the West, most of the compiling technology assessed dealt with vectorizers/parallelizers, system software that analyzes sequential programs and produces executable code for vector or parallel processors, either in an extended version of the original source language (source-to-source translators or pre-compilers) or in something closer to the machine code of a computer with parallel processing capabilities (vectorizing compilers). The Soviet vectorizing systems generally use methods developed in the United States, enhanced or extended by their own methods, which are claimed to be new. Several operational systems have been produced. The overall volume of work reported is substantially less than in the West and there was no evidence of any important Soviet breakthroughs. It is difficult to make performance comparisons because the hardware platforms are so difficult to calibrate, but the speed-ups reported are generally in line with Western experience. The reported systems often have a "first generation" character to them, suggesting that there is no pressure from users to develop more refined or more sophisticated techniques.

There has been Soviet work on most varieties of parallel languages considered in the West, including extended languages, coarse-grain composition languages, functional languages, asynchronous process languages, and languages suitable for fine-grain parallelism, though the number of instances of each is fewer than in the West. The languages are generally copies, variants, or extensions (often substantial) of Western languages. In a few cases, truly unique new languages have been developed, with MAYaK, Polyar, and BARS being perhaps the most interesting. A striking characteristic of most of the Soviet language studies is that no implementation details or performance numbers are reported. This makes it extremely difficult to assess how effective the research has been. Generally, the work has parallels in the West that seem at least as successful.

In line with long-standing Soviet programming language research traditions, designers of most Soviet parallel processing languages emphasize the principles on which their design is based or formulate much of their exposition in terms of theoretical foundations. It is impossible to determine whether this use of theory has been empowering or limiting, because there has been so little experience with the resulting languages.

One programming language topic in which the Soviet Union has been competitive with the West is coarse-grain composition languages. These process composition languages are used to assemble modules written in (possibly different) base procedural languages into a single parallel computation. The capabilities offered by Soviet coarse-grain composition languages are quite comparable to those available in Western languages.

Soviet parallel programming languages and systems have advanced quite far, considering the limited availability of parallel hardware. If Western or Japanese computers become available to fill this void, there is a small group of well-trained researchers in the successor states of the former Soviet Union who could put them to effective use. Given a choice, these researchers are likely to adopt the larger, more stable base of Western languages and support software for their subsequent research rather than port Soviet products to the new machines.

ALGORITHMS AND APPLICATIONS

Research on parallel algorithms falls into two general categories. The first is concerned with how best to parallelize existing sequential algorithms so that they can be executed efficiently on a parallel computer. The second is concerned with creating new or redesigned algorithms, more suitable than present ones for efficient execution on a parallel machine. Over the years, adoption of more efficient (in the sense of requiring less or easier computation) algorithms and parallelization have contributed comparably to speeding up the most demanding numerical computations.

Assessment of published research literature in parallel algorithms suggests that researchers in the former Soviet Union have been pursuing lines of investigation very similar to those pursued in the United States and Europe. These areas include core algorithms applicable to many fields of physical modeling and analysis, such as linear algebra and fast Fourier transforms (FFTs); algorithms for systolic architectures; algorithms for signal processing; algorithms for computational fluid dynamics; and algorithms for boundary value problems. With a few exceptions, however, Soviet accomplishments have lagged behind those of the United States and Europe, typically by five to 10 years. In a few areas, notably domain decomposition and some

aspects of linear algebra, Soviet parallel algorithm research has been much closer to parity with work in the West.

In certain areas of applied mathematical research related to parallel algorithms, Soviet work has been first-rate, sometimes superior to that in the West, and has been eagerly assimilated by Western parallel algorithm developers. A number of approaches that are now very important in Western parallel algorithm development were first conceived, analyzed, and documented in the Soviet Union. They were only fully investigated, understood, and implemented as working computer programs years later, in the West. In these cases, lack of computing hardware to develop and implement these ideas fully clearly retarded Soviet progress.

The Soviet lag in high-performance computers has put Soviet computer scientists and mathematicians at a substantial disadvantage in parallel algorithm research. Emphasis on (and skill at) analytical modeling and error analysis cannot fully compensate for lack of the high-performance computers that allow Western and Japanese researchers to quickly experiment with and test computational ideas. This situation is likely to change when inexpensive parallel processing computers based on Western processors become available to parallel algorithm developers in the successor states of the former Soviet Union, a process that has already begun.

CHAPTER I
INTRODUCTION AND OVERVIEW OF ASSESSMENTS

This technical assessment report examines strengths and weaknesses of parallel processing research and development in the Soviet Union during the 1980s (through the first half of 1991). This work may continue in the successor states that emerged at the end of 1991 if support for the work is continued by the new governments. It is the only basis available for any efforts to develop parallel processing technology the new nations may choose to pursue.

A. INTRODUCTION

Parallel processing research aims to develop and improve computing hardware and software that uses multiple processing units to increase the overall speed of demanding computations, by executing as many operations as possible simultaneously rather than sequentially. This strategy for acceleration of computation is being pursued in the hope that it will evade present physical and engineering speed limits for conventional sequential (von Neumann) computers. To some extent, that hope has already been realized in the fastest computers commercially available in the West, Japan, and (to a much more limited extent) the former Soviet Union. Contemporary research seeks to push the bounds back further by realizing more fully the benefits of parallelism.

Obstacles to use of parallelism as a strategy for accelerating computation are of two kinds: limits imposed by the fact that some parts of every problem must be solved in sequence, and limits imposed by the fact that preparation for parallel computation and communication between processes executing in parallel themselves consume some of the time parallelism is intended to save. Parallel processing research is therefore concerned with identifying those problems for which the necessarily sequential portion is potentially small, and devising effective and efficient ways of extracting and carrying out the parts of the computation that can be accelerated by use of multiple processors acting in concert. The goal is automatic preparation of powerful algorithms for machines that make optimal use of many processors, minimizing the overhead cost of necessary communication.

Many paths toward this goal have been and are being explored. Some basic parameters that designers of parallel computing systems must consider are:

- the number of instruction streams and the number of data streams with which the system deals simultaneously. The classical von Neumann computer executes a single instruction stream on a single data stream and is designated an SISD computer in one popular classification scheme. In a single instruction stream/multiple data stream (SIMD) computer, the same procedure is applied simultaneously to multiple data sets by separate (generally identical) processors. In a multiple instruction stream/multiple data stream (MIMD) computer, a number of more-or-less independent (and possibly different) processors each applies its own procedure to its own data stream.

- the nature, frequency of operation, and topology of the communication and scheduling/synchronization mechanisms that allow processors to cooperate in carrying out a complex computation. One important aspect of this is the granularity (or grain size) of the computer architecture: whether the basic division of labor takes place at the level of entire procedures, which individual processing units must be powerful enough to execute (coarse granularity or task-level parallelism); at the level of simple, primitive machine instructions that can be executed by very simple processing units (fine granularity or data-level parallelism); or at an intermediate level (medium granularity or operation-level parallelism). In tightly coupled systems, shared memory and a shared clock keep different processors in close synchronization, logically and temporally; in loosely coupled systems, processors exchange information (including timing information) by explicit communication (usually involving some kind of handshaking). Communication between processors (implicit or explicit) can be direct (if the communicating processors are wired to one another), indirect (if other processors carry the message), or switched (if communication is via dedicated communication hardware, such as a bus).

- the control principles that organize computation. Systems with centralized control over instructions and data are called control-driven. If the tasks under control are mutually independent, they can be carried out synchronously; if they must be explicitly coordinated, provision must be made for

asynchronous control/communication. Systems with decentralized control over instruction execution can be data-driven or demand-driven. In data-driven computations, instructions are executed as soon as all required input data are available; the input data values are consumed as the instructions are executed. In demand-driven computations, instructions are executed only when their products are needed by an instruction further along in the computation.

Table I.1 indicates some of the combinations of design parameters that have been embodied in commercial or experimental parallel processing computers.

An analogy with music may help the reader avoid a trap that has ensnared some observers in the past: that the magic of parallelism can easily and straightforwardly turn plentiful, cheap processors into powerful supercomputers. Individual instruments are like processors. Over the course of time, they have evolved from simple pipes, drums, and lyres into more complex, powerful, and technologically advanced instruments like organs, pianos, and Stradivarius violins. Parallel computation is like symphonic music. Conceiving appropriate music, orchestrating that music so that it makes best use of each instrument, and communicating with the various musicians (before and during the performance) to achieve the desired complex sounds, rhythms, and harmonies are extraordinarily difficult tasks that few have mastered. Genius has been a key element in development of the symphony, but at every stage that genius was grounded in deep understanding (and good use) of the latest instrumental and performance technology and of the hard-won insights of previous generations of symphonic composers and conductors. There was not and could not have been "leapfrogging" from primitive folk tunes to Bach, Mozart, and Beethoven, and we should not expect to see powerful parallel computers developed by simply skipping difficult steps in the development process. And we should not expect to see complex, powerful computations performed by analogs of an orchestra of pan pipes.

Table I.1
PARALLEL PROCESSING TAXONOMY

Architecture	Category	Control/ Communication Principle	Connection/ Communication Topology	Coupling Mechanism	Granularity	Remarks
Multi-processor	MIMD	Asynchronous, by global operating system	Simple	Bus (to shared memory, I/O & mass storage)	Task level (coarse)	Semi-independent processors
Vector Processor	SIMD	Synchronous	Linear array	Tight, hardwired; shared memory	Operation level (medium)	Host computer peripheral
Pipeline Processor	SIMD	Synchronous	Linear array	Moderate, hardwired	Data/ operation level (fine-medium)	Simultaneous "sequential" executions
Array Processor	SIMD	Synchronous	2-D array, cylinder, or hypertorus	Tight, hardwired	Data level (fine)	Host computer peripheral
Systolic Processor	SIMD	Synchronous (global clock)	Possibly complex array	Tight, local, hardwired	Data level (fine)	Host computer peripheral
(Hyper)Cube	MIMD (or SIMD)	Asynchronous (or synchronous)	Hypercube edges	Message-passing, local	Data level (fine)	Local memory only
Butterfly	SIMD	Synchronous	2-D array; vertical, oblique connections	Tight, hardwired	Task level (coarse)	FFT specialist
Dataflow	MIMD	Asynchronous dataflow	Simple	Loose-moderate, shared data	Operation level (medium)	Local memory only
Connection Machine	SIMD	Synchronous dataflow	Programmable	Local + global	Task level (coarse)	Dynamically reconfigures

B. GENERAL OBSERVATIONS

Soviet computer research and development organizations have pursued many of the major avenues of inquiry related to parallel processing that their colleagues in the West have chosen to explore. However, the limited number of people and groups engaged in this effort in the Soviet Union and the breadth of the Soviet effort (and possibly a strong political/economic/social disinclination to avoid duplication of effort) have limited the collective depth of Soviet parallel processing research. Little or no overlap was observed between activities at different institutes, severely limiting opportunities for cross-fertilization or expert constructive criticism.

The ideas embodied in Soviet parallel processing hardware and software have often started the Soviet development cycle at the leading edge of research. However, limited ability to build, demonstrate, and test embodiments of these ideas has delayed and limited Soviet achievement. The inability of Soviet industry to build sufficiently powerful computers and computer components has been particularly limiting. Lifting or relaxation of past/existing restrictions on Soviet successor Republic import of powerful hardware could dramatically change this situation, if parallel processing researchers there can obtain financial support for purchase of foreign machines, components, and software tools. Whether they will be able to command such support is unclear. Aside from short-term uncertainties associated with the present (winter 1991-92) economic near-anarchy in the successor states of the former Soviet Union, the desire and ability of the successor states to support long-term research like parallel processing development have yet to be established. In addition, much of the former military justification and support for indigenous advanced computer development has evaporated over the past few years.

For several decades prior to the 1980s, development and production of Soviet mainframe and mini-computers were dominated by efforts to build reverse-engineered indigenous machines that were functionally similar to selected Western computers. There was no analogous concentration of effort on reverse-engineering of specific Western systems in the work of Soviet parallel processing researchers in the 1980s. The quality of the research assessed by the panel varied from very sound and interesting to pedestrian, with most of the groups at the major hardware and software centers to which the work is largely confined doing good (or at least serious) research. In a few instances, interesting and competent parallel language develop-

ment work was found at institutions not associated with hardware development efforts.

No evidence was found of successful Soviet attempts to use breakthroughs in parallel processing technology to "leapfrog" impediments and limitations that Soviet industrial weakness in microelectronics and other computer manufacturing areas impose on the performance of high-end Soviet computers. As was noted above, such leapfrogging cannot be expected because highly capable parallel processing hardware and software can only be built on a strong foundation in the sequential processing technology base, with a qualitatively higher level of complexity. However, there are numerous examples of sound, creditable Soviet parallel processing research. The sections and chapters that follow describe some of the more interesting Soviet efforts in parallel processing hardware development, parallel processing software, and parallel computing applications.

C. HARDWARE

Soviet development of mainframe, mini-, and micro-computers was characteristically imitative, striving to produce functional equivalents of specific Western computer series. By contrast, Soviet attempts to develop parallel processing machines have shown a considerable degree of originality. Only one high-performance Soviet computer, the vector-pipelined Elektronika-SSBIS (the "Red Cray"), can be considered a close analog of a Western machine.

The El'brus and PS-2x00 projects, carried out within industrial ministries, show the inspiration of Western developments. These were mid-term industrial efforts to produce multiprocessor parallel machines to improve Soviet high-performance computing capabilities. High-level Western design ideas were generally followed, but in each case some features of the implementation were indigenous and in several cases problems in Western designs were overcome.

The El'brus machines—designed by the Precision Mechanics and Computer Technology Institute of the USSR Academy of Sciences[1] (ITMVT) in Moscow—are

---

1 At the end of 1991, the USSR Academy of Sciences changed to the Russian Academy of Sciences, the name it used before June 1925.

successors to that institute's successful BESM-6, which was the workhorse of Soviet scientific computing for two decades. Representing the leading edge of Soviet development work on coarse-grain few-processor systems, the El'brus machines are based on Soviet state-of-the-art components from the Ministry of Electronics Industry, which the El'brus design group stimulated to produce better components. A distinctive feature of these machines is their hybrid hardware/software provisions for error/failure tolerance (included to compensate for the expected unreliability of processors and other components). About 50 El'brus-2 machines were installed in the second half of the 1980s, with serial production scheduled to continue. Development of the El'brus-3, a world state-of-the-art very-long-instruction-word (VLIW) machine based on Soviet emitter-coupled logic (ECL) gate arrays, began in 1987 and is scheduled to produce prototypes in 1992. If the El'brus-3 goes into serial production within the next few years, it will be the only commercially available VLIW computer in the world.[2]

The Reconfigurable System (PS-2000) series machines were developed by the "Impul's" Scientific Production Association and the Control Problems Institute in Moscow, at the same time the El'brus machines were being developed. The PS-2000 designs were based on then-existing Soviet components, with design features to compensate for the known deficiencies of those components (as with the El'brus machines, especially with regard to reliability). About 200 PS 2000 SIMD machines were installed in Soviet scientific and technical institutions in the 1980s. The PS-2100, a multi-SIMD machine with semiconductor external storage (and floating point hardware implemented in gate arrays), began shipping in the late 1980s. Users report that their PS-2100s can be up to 10 times faster than the PS-2000s they replaced. A PS-2300 design effort began at the beginning of the 1990s.

Soviet development work on multiprocessor systems with replicated processors began in the 1960s. There were half a dozen major projects of this nature in the Soviet Union in the 1980s. These included: the work on dynamic architecture machines of V. A. Torgashev; the macro-pipeline work of I. N. Molchanov, A. A. Letichevskiy, and others at the Glushkov Cybernetics Institute in Kiev; the program-

---

2 Note added in proof: At the end of February 1992, Sun Microsystems announced a contract with the El'brus advanced development team to support software development work of interest to Sun by about 50 members of the team in Moscow.

mable architecture machines at the Multiprocessor Computer Systems Scientific Research Institute (NIIMVS) in Taganrog; the Kronos projects under V. Ye. Kotov in Novosibirsk; work by N. N. Mirenkov at the Mathematics Institute of the Siberian Branch of the USSR/Russian Academy of Sciences in Novosibirsk; and work at the Semiconductor Electronics Institute under Khoroshevskiy, also in Novosibirsk. Several of these longer-term many-processor parallel processing machine design and evaluation efforts began in the early 1980s, under the sponsorship of the research and development arm of the manufacturer of Unified System (ES) mainframes.

These projects include research in most major categories of parallel architecture, including coarse-grain shared memory systems with few processors, SIMD machines, dataflow and/or reduction machines, and reconfigurable architectures. Thus, Soviet parallel hardware research has shown good breadth. However, the small number of groups involved in the effort means the depth produced by multiple research groups pursuing similar ideas is generally absent.

These groups have done their work on what would be considered unusually long development cycles in the West. Many of their ideas, representing the world state of the art when they were conceived, were no longer so by the time they were realized in functioning hardware. Long life-cycles also hindered rapid building of prototypes and experimentation, which have been crucial to the development of parallel systems in the West.

The inter-processor interconnect schemes represented in the products of these research efforts have shown limited variety, leaning mostly toward switched crossbar or network configurations. Only one example of a hypercube topology was seen—and this only recently—and no examples of butterfly interconnection. The one hypercube connection experiment, in a neural-network architecture at NIIMVS in Taganrog, probably indicates that researchers there are beginning to work seriously on massively parallel (>1,000 processors) systems.

The core project at NIIMVS has long been the Multiprocessor Computer System with Programmable Architecture, centered on programmable crossbar interconnection and modified dataflow control ideas. The Taganrog researchers produced a preliminary prototype machine, the ES-2703, in 1985. In the early 1990s, NIIMVS has won major competitive grants for the development of neural network systems, and

of a "supermacrocomputer" embodying ideas from that institute's traditional approach to reconfigurable architecture, artificial intelligence, and component development. Complementary projects at NIIMVS include efforts to develop signal-processing ("problem-oriented") computers; neurocomputers (an effort that has produced a single-neuron "neurochip," asserted to be highly networkable in hypercube topology); and robotics. In the early 1990s the Taganrog group seems to have become the largest and most dynamic many-processor research/design effort.

Although difficult to evaluate precisely, Soviet research on dataflow and reduction machines has reflected some ideas and approaches not found in Western machines. One of the most notable examples of this is the Dynamic Architecture Machine (MDA) project at the Informatics and Automation Institute in St. Petersburg. These researchers have been exploring a novel "dynamic automata network" control scheme, which seems to support mixture or dataflow, control flow, and demand driven paradigms in the course of a computation. This group produced a few prototypes (designated ES-2704) incorporating some of their flexible-control ideas in the mid-1980s; these machines were also vehicles for the group's RYaD and RL languages. A follow-on machine, the ES-2727, is said to be under development, but the current status of that project is not clear.

All Soviet parallel processing hardware development has been restrained by an inadequate component base in Soviet industry. Even something like fast external storage, taken for granted in the West and Japan, has often been unavailable to Soviet researchers. For example, disks with 317 Mbytes of storage are the largest available, but are in very short supply. At least one high-performance computer manufacturer has been forced to resort to the use of personal computer hard-disks. We have seen only one example (in the PS-2100) of solid-state external storage, which is critical to the high performance of many Western supercomputers.

Some major Soviet parallel processing hardware projects have suffered from pushing too many Soviet technological boundaries at once. For example, the El'brus projects have attempted to develop new architectures, components, subsystems, CAD systems, and manufacturing technologies simultaneously. Shortfalls in the individual areas have delayed the overall effort. In addition, systems built around a shaky advanced component base have been predictably unreliable. Architecture fea-

tures designed to provide redundancy and fault-tolerance have been only partially successful in overcoming this weakness.

Relief from component reliability problems began to appear in the mid-1980s, when use of new, relatively reliable, relatively high-performance Soviet gate arrays led to significant improvements in performance and reliability of new Soviet multiprocessor computers.

This trend is likely to accelerate as powerful, reliable Western components become available to developers of parallel computer systems in the successor states of the former Soviet Union. At the beginning of the 1990s, powerful Western processors such as the Intel 386, 486, and i860 microprocessors, the Motorola 68030 and 68040 microprocessors, and Inmos transputers began to become available for rubles within the Soviet Union. Use of these processors will allow developers of parallel processing hardware to evade the shortcomings of indigenous microelectronics and manufacturing capabilities, shortening development cycles and placing reliable systems into the hands of software developers and users quickly. In the past, the products of Soviet multiprocessor research rarely reached software developers in the form of reliable, serially produced machines. In the future, availability of parallel computing hardware based on Western components could improve dramatically the conditions under which parallel processing applications and software are developed in the states of the former Soviet Union.

European emphasis on transputers (processors manufactured by the British company Inmos that have built-in interprocessor communication links) seems to be producing a strong echo in parallel processing research in the successor states of the former Soviet Union. (Relatively easy availability and relatively low cost to the developer are probably influential factors in both research communities.) Nearly all of the major Soviet parallel processing groups discussed above, which have traditionally designed their processors using the technology base that was available in the Soviet Union, reportedly are seeking to implement some of their machine design and software ideas on transputer platforms. These groups claim to have several tens of real transputers each, with which they are experimenting. The most powerful current transputer processor (the Inmos T800, the 20 MHz version of which runs the Linpack Benchmark in 0.4 MFLOPS, equivalent to a Sun 3 processor) is still under export controls. Older, less powerful TRAM (transputers plus associated memory)

units are available in rubles for approximately the same amount as a PC/AT personal computer.

The availability of Western components has allowed new groups to become involved in parallel computer development. For example, International Center for Informatics and Electronics (InterEVM) researchers are building several multiprocessor systems based on Western processors. These include four-processor bus-interconnected systems using Intel i860 or MIPS R3000 RISC (reduced instruction set computer) processors, and accelerator boards for PCs that incorporate transputers. These systems do not necessarily offer new scientific ideas, but are probably forerunners of more ambitious examples of the penetration of powerful Western components into the successor states of the former Soviet Union. The "Kvant" Scientific Research Center and the Informatics Problems Institute in Moscow are also actively involved in developing transputer-based systems for their own applications. Researchers at the (Keldysh) Applied Mathematics Institute in Moscow report that their locally-built eight-processor transputer system solves some fluid dynamics problems significantly faster than on their most powerful ES mainframes or El'brus scientific computers.

Soviet scientific or mathematical researchers have published a handful of designs or descriptions of application-specific processors and multiprocessors. Although limited to very narrow applications domains, such systems can sometimes provide very high performance at moderate cost, using components and subsystems that are in indigenous series production. For example, a processor tailored to Monte Carlo calculations for Ising spin lattices was so successful that Western scientists are reported to be building upgraded versions of this machine (using Western components). Other published Soviet application-specific processor designs have included a systolic processor for solving systems of linear equations, a processor for magnetohydrodynamics problems capable in principle of running at 1 GFLOPS, and a processor for detecting particles by momentum in physics experiments.

Economic troubles in the successor states of the former Soviet Union have harmed all the major parallel processing hardware projects. Support for fundamental research in high-performance computing from the State Committee on Science and Technology, the State Committee on Higher Education, and their successors has remained relatively constant over the last few years, although the true value of this

support has been eroded by accelerating inflation. Several hardware development projects have been terminated, and those that remain are having increasing problems getting support for research and manufacturing.

The market for expensive, high-performance computers in the successor states of the former Soviet Union has weakened significantly in recent years, making industry reluctant to manufacture the systems, components, and subsystems needed to develop future high-performance parallel processing machines. In a number of projects, researchers have responded by trying to apply their design ideas to the development of small-scale systems that can be attached to personal computers and workstations.

D. PROGRAMMING

Conventional computer programming specifies a sequence of steps that will solve a problem. For parallel computation, it is also necessary to determine how these steps can be divided up among independent processors and applied to the common data without "unexpected collisions." Exponential explosion of the number of potential interactions qualitatively changes programming, generally precluding pre-planning of computations by the programmer at the lowest level of detail. The complexity of programming, already substantial in the sequential case, increases dramatically. Complexity-reducing tools, abstractions and methodologies are essential.

Parallel programming research invents and evaluates new languages, language constructs, and programming tools (including compilers) needed for expressing the new features and carrying out the new tasks. These are often integrated into programming environments (including debuggers, verification aids, and easy-to-use interfaces with conventional languages) designed to simplify the parallel programming task.

Soviet research in parallel programming has addressed many of the same topics that are actively pursued in the West and Japan, though measured by the number of active research groups or the number of research publications, fewer scientists seem to be working on these problems. Generally, the Soviet work has extended Western work rather than broken new ground. Soviet research groups have been active in

automatic vectorization/parallelization (via translators that attempt to discover and exploit parallelism automatically); in parallel programming languages (which attempt to make the parallel programming task easier, use a parallel architecture effectively, or both), and (to a more limited extent) in development of parallel programming environments and tools.

In several instances the Soviet work has been of high quality, though none of this work is superior to comparable efforts in the West and Japan. In most cases, developers of Soviet parallel processing software have apparently had no access to working parallel hardware. This limited experience with actual hardware is evident in the content of the work, and experimental results are seldom described. Perhaps conversely, more explicit attention has been devoted to mathematical foundations than is usual in the West or Japan. This analytic sophistication and the quality of some of the work (despite its limitations) suggest that several groups could rapidly make good use of reliable parallel hardware, were it to be made available to them.

As in the West, most of the compiling technology assessed dealt with vectorizers/parallelizers, system software that analyzes sequential programs and produces executable code for vector or parallel processors, either in an extended version of the original source language (source-to-source translators or pre-compilers) or in something closer to the machine code of a computer with parallel processing capabilities (vectorizing compilers). This has clearly been an active research area in the Soviet Union.

The Soviet vectorizing systems generally use methods developed in the United States, enhanced or extended by their own methods, which are claimed to be new. Though published information (in both communities!) is often insufficient to verify these claims, the nature of the area suggests that some of these claims are almost certainly true. Several operational systems have been produced. The overall volume of Soviet work reported has been substantially less than in the West, and there was no evidence of any important Soviet breakthroughs; as in most work everywhere on this topic, the advances are incremental.

It is difficult to make performance comparisons because the hardware platforms are so difficult to calibrate, but the speed-ups reported have been generally in line with Western experience. The component most strikingly missing from the Soviet

vectorization literature has been extensive, long-term experience with operational systems applied to everyday problems. The reported systems often have had a "first generation" character to them, suggesting that there has been no pressure to develop more refined or more sophisticated techniques. This has probably been due to the lack of a substantial user community with routine access to vectorized supercomputers.

As in the West, most Soviet vectorizers/parallelizers work with FORTRAN source code provided by the programmer. One example, the Fora-ES Converter produced by the Keldysh Applied Mathematics Institute in Moscow, analyzes FORTRAN IV for opportunities for parallel execution, producing a new source program that uses extensions of that language to move part of the computation to a special-purpose vector processor. This source-to-source translator contains a few modest extensions of Western practice, and was able to discover vectorization possibilities in six of 14 Livermore Loops (standard benchmarks) against which it was tested. Speed-ups of 1.2 to 8 times were claimed relative to execution on the mainframe without the vector processor.

A second instructive example of Soviet vectorizers/parallelizers is the Vector Pipelining Compiler developed by the Cybernetics Problems Institute in Moscow. This compiler vectorizes PL/1 array commands and simple inner loops (meeting stringent conditions) for the Elektronika-SSBIS vector pipelining computer. It could vectorize four of 14 of the Livermore Loops against which it was tested, producing a claimed speedup of five to 10 times over sequential computation. As in all such comparisons, it is difficult to assess how much of this advantage came from local innovation in the vectorizer and how much simply came from the speed advantage of the vector processing hardware.

It has been possible to find instances of Soviet work on most varieties of parallel languages considered in the West, including extended languages, coarse-grain composition languages, functional languages, asynchronous process languages, and languages suitable for fine-grain parallelism, though the number of instances of each is fewer than in the West. The languages are generally copies, variants, or extensions (often substantial) of Western languages. In a few cases, truly unique new languages have been developed, with MAYaK, Polyar, and BARS being perhaps the most interesting. A striking characteristic of most of the Soviet language studies is that no

implementation details or performance numbers have been reported. This makes it extremely difficult to assess how effective the research has been. In no case was the Soviet work more successful than parallel Western efforts.

In line with long-standing Soviet programming language research traditions, designers of most Soviet parallel processing languages emphasize the principles on which their design is based or formulate much of their exposition in terms of theoretical foundations. This is uncommon in the West, except in functional programming. It is impossible to determine whether this use of theory has been empowering or limiting, because there has been so little experience with the resulting languages.

Extended languages for parallel processing add constructs to control parallel execution, such as *parbegin* and *parend*, to a sequential language. In the former Soviet Union, as elsewhere, that language is most often FORTRAN. Russian researchers at the Multiprocessor Computer Systems Scientific Research Institute (NIIMVS) in Taganrog have also produced Pascal parallel processing extensions for their ES-2703 prototype parallel computer, and workers at the Siberian Branch of the USSR/ Russian Academy of Sciences in Novosibirsk have even invented an extended version of COBOL (p-Kobol).

One programming language topic in which the Soviet Union has been competitive with the West is coarse-grain composition languages. These process composition languages are used to assemble modules written in (possibly different) base procedural languages into a single parallel computation. The capabilities offered by Soviet coarse-grain composition languages are quite comparable to those available in Western languages.

Though experimental results are very limited, the two most significant Soviet coarse-grain composition systems seem to be Kiev State University's Parallel Asynchronous Recursive Controllable System (PARUS), and the Macropipelined Language (MAYaK) system developed by the Glushkov Cybernetics Institute in Kiev. PARUS is a set of tools for extending base languages like Pascal or C, so programs written in those languages can be combined into parallel computations. MAYaK, based on Soviet theoretical work on data structures, discrete system design, and macropipelining, provides a somewhat novel application programmer's high-level language, a static macropipeline assembler, and a dynamic control language for

multi-macropipeline computations. For one 15,000-line "industrial" program, the Glushkov researchers claimed a 33-times speedup on their ES-2701 parallel processor prototype relative to their high-end conventional ES-1066 mainframe.

Soviet research on functional languages, asynchronous process languages, and languages suitable for expressing fine-grain parallelism seems to have been carried out on a smaller scale, and is less notable, than the parallel processing language work discussed in the preceding paragraphs. FPL, SFP, RYaD, and RL are some of the Soviet functional languages (languages based on a functional formulation of algorithms, with parallelism implicitly captured in argument evaluation). Polyar, developed by the Siberian Branch of the USSR/Russian Academy of Sciences in Novosibirsk as part of the START Project, is the most notable Soviet asynchronous process language (a language supporting the concurrent sequential processes model of parallel computation). Working Polyar software systems exist, but the Novosibirsk researchers have reported no results from execution on actual parallel hardware. BARS, developed by a different group at the Siberian Branch of the USSR/Russian Academy of Sciences in Novosibirsk, is the most notable Soviet system suitable for expressing fine-grain parallelism (common operation sequences to be performed uniformly across data, in parallel). Soviet activity in this area has been very limited.

Soviet parallel programming languages and systems have advanced quite far, considering the limited availability of parallel hardware. If Western or Japanese computers become available to fill this void, there is a small group of well-trained researchers in the successor states who could put them to effective use. Given a choice, researchers in the successor states are likely to adopt the larger, more stable base of Western languages and support software for their subsequent research rather than port Soviet products to the new machines (though porting the Soviet software probably would not be difficult).

E. ALGORITHMS AND APPLICATIONS

Research on parallel algorithms falls into two general categories. The first is concerned with how best to parallelize existing sequential algorithms so that they can be executed efficiently on a parallel computer. The second is concerned with creating new or redesigned algorithms, more suitable than present ones for efficient execu-

tion on a parallel machine. Generally, algorithms parallelize best when they are derived from mathematical analysis that decomposes the problem space into simpler parts, each of which can be treated separately using a relatively simple procedure; a solution of the original problem is then obtained by combining the partial solutions in a relatively straightforward way. Over the years, adoption of more efficient (in the sense of requiring less or easier computation) algorithms and parallelization has contributed comparably to speeding up the most demanding numerical computations.

Assessment of published research literature in parallel algorithms suggests that researchers in the former Soviet Union have been pursuing lines of investigation very similar to those pursued in the United States and Europe. These areas include core algorithms applicable to many fields of physical modeling and analysis, such as linear algebra and fast Fourier transforms (FFTs); algorithms for systolic architectures; algorithms for signal processing; algorithms for computational fluid dynamics; and algorithms for boundary value problems. With a few exceptions, however, Soviet accomplishments have lagged behind those of the United States and Europe, typically by five to 10 years. In a few areas, notably domain decomposition and some aspects of linear algebra, Soviet parallel algorithm research has been much closer to parity with work in the West.

In certain areas of applied mathematical research related to parallel algorithms, Soviet work has been first-rate, sometimes superior to that in the West, and has been eagerly assimilated by Western parallel algorithm developers. A number of approaches that are now very important in Western parallel algorithm development were first conceived, analyzed, and documented in the Soviet Union. They were only fully investigated, understood, and implemented as working computer programs years later, in the West. In these cases, lack of computing hardware to develop and implement these ideas fully clearly retarded Soviet progress.

Perhaps the best example of this is the multigrid method, invented by N. S. Bakhvalov in the mid-1960s. Each operation of a multigrid calculation is local, considering and affecting only grid points that are nearest neighbors; consequently, each operation is a candidate for parallelization. Multigrid methods were originally applied to simple boundary value problems, which arise in many physical applications. More recent applications include network problems, structural problems, image processing, control theory, combinatorial optimization (the traveling salesman

problem), statistical mechanics (the Ising model), and quantum electrodynamics. Ironically, these techniques found their way back into Soviet computer mathematics textbooks once they were developed for parallel processing in the West.

The story has been different (and happier for Soviet scientists) in another important area of mathematics related to parallel computation, domain decomposition. Yuriy A. Kuznetsov at the Scientific Research Computer Center of the USSR/Russian Academy of Sciences in Moscow made significant contributions to the development of this widely applicable technique in the course of his research on electromagnetics calculations in 1984. His subsequent work has concentrated on proving convergence theorems, and his implementations of these naturally parallel algorithms have been (necessarily) on small scalar machines. However, he is respected as a peer by many US and European researchers active in this area; his work is considered state-of-the-art. Others Soviet researchers, including V. I. Agoshkov (Moscow), A. M. Matzokin (Novosibirsk), and Sergey Nepomnyaschikh (Novosibirsk), have also actively contributed to the advancement of domain decomposition.

Often, however, the Soviet lag in high-performance computers has put Soviet computer scientists and mathematicians at a substantial disadvantage in parallel algorithm research. Emphasis on (and skill at) analytical modeling and error analysis cannot fully compensate for lack of the high-performance computers that allow Western and Japanese researchers to quickly experiment with and test computational ideas. This situation is likely to change when inexpensive parallel processing computers based on Western processors become available to parallel algorithm developers in the successor states of the former Soviet Union, a process that has already begun.

One example that may be typical of emerging developments is a solicitation of collaboration by a group of researchers at the Computer Center of the Russian Academy of Sciences' Siberian Branch in Novosibirsk who are actively working with a transputer-based computer. Recently, they posted a number of active projects and solicited contacts and collaborations with Western scientists. Their interests include parallel program synthesis, software tools for fine-grain parallel programming, design and analysis of systolic/wavefront algorithms, algorithms for structure synthesis and subsystems placement for multi-transputer systems, parallel program

mapping on parallel architectures, debugging systems for parallel programs, and algorithms and software for modeling electrophysical and fluid dynamics problems.

(blank)

CHAPTER II
PARALLEL HARDWARE

A. SUMMARY

- Soviet parallel hardware research has shown good breadth, but little depth. While it is possible to find examples of Soviet research in most major categories of architectures—for example, coarse-grain shared memory systems with few processors, single-instruction multiple data, dataflow/reduction, reconfigurable architectures—there are rarely multiple research groups pursuing similar ideas.

- In sharp contrast to the development of mainframe, mini-, and micro-computers, Soviet parallel machines show high levels of originality. Only one high-performance computer can be considered an analog of a Western model. This is the vector-pipelined Elektronika-SSBIS. None of the parallel processors have close Western analogs.

- Several projects, especially the El'brus and PS-2x00 projects that were developed within industrial ministries, show the inspiration of Western developments, however. Certain high-level Western ideas have been adopted, but in each case the implementation is indigenous. In some cases, the implementations improve on the Western work.

- Some major Soviet projects suffered from pushing too many technological boundaries at once. The El'brus projects, in particular, require the development of new architectures, components, subsystems, CAD systems, and manufacturing technologies simultaneously. As a result, development is very difficult and time consuming.

- Systems using an advanced component base suffer from reliability problems. Architectures designed to provide redundancy and fault-tolerance have been only partially successful in overcoming this weakness.

- The appearance of gate arrays during the mid-1980s led to significant improvements in machine performance and reliability.

- Because of long life-cycles, some ideas, while state of the art when they were conceived, had fallen behind comparable Western hardware by the time they were realized in a functional machine. Long life-cycles also hinder rapid building of prototypes and experimentation, which is crucial to the development of parallel systems.

- Soviet research on Very-Long-Instruction-Word (VLIW) architectures is currently at, or very close to, the world-wide state of the art.

- Although difficult to evaluate precisely, Soviet research on dataflow/reduction machines reflects some ideas and approaches not found in Western models and should be investigated more closely.

- Economic crises in the former Soviet Union have harmed all hardware development projects: several have been terminated, and those that remain are having increasing problems getting support for research and manufacturing. The market for expensive, high-performance machines has weakened significantly in recent years; it is therefore increasingly difficult to convince industry to manufacture the necessary systems, components, and subsystems.

- In a number of projects, researchers are applying existing design ideologies to the development of small-scale systems that can be attached to personal computers and workstations. This trend also reflects the new realities of the high-performance computer market.

- Support for fundamental research in high-performance computing from the State Committee on Science and Technology and the State Committee on National Education has remained relatively constant over the last few years, although the true value of this support has been eroded by inflation and shortages of materials, components, and subsystems. The Multiprocessor Computer Systems Scientific Research Institute in Taganrog has won major grants for the development of neural network systems, and a "supermacrocomputer" that incorporates that institute's traditional approach to reconfigurable architectures, artificial intelligence, and component development.

- All systems suffer from inadequate external storage. Disks with 317 Mbytes (Mb) of storage are the largest available, but are in very short supply. At least one manufacturer is resorting to the use of personal computer hard-disks.

- We have seen only one example (in the PS-2100) of solid-state external storage. Such storage is critical to the high performance of many Western supercomputers.

- The inter-processor interconnect schemes show limited variety. We have seen only one example of a hypercube topology—and this only recently—and have no examples of a butterfly interconnect. The one use, in a neural-network architecture at the Multiprocessor Computer Systems Scientific Research Institute in Taganrog, probably indicates that researchers here are beginning to work seriously on massively parallel (>1,000 processors) systems.

- Western components are likely to play a significant role in the development of parallel systems in the successor states of the former Soviet Union. Over the last two to three years, powerful Western processors such as the Intel 386, 486, and i860 processors, the Motorola 68030 and 68040, and Inmos transputers became available for rubles within the Soviet Union. In our estimation, a significant number, perhaps most, of the major parallel computer projects are now experimenting with such processors, particularly transputers. By using such processors and avoiding dependency on indigenous microelectronics and manufacturing capabilities, researchers in the successor states can shorten life-cycles and more easily share results and transfer technology to users.

- Several application-specific parallel computers have been publicized within the last three to four years. Although limited to very narrow applications domains, such systems can potentially provide very high performance at moderate cost and with components and subsystems in series production.

B. INTRODUCTION

The Soviet Union had a long history of research and development (R&D) in digital computing. The first Soviet digital, stored-program computer, the MESM (Small Electronic Calculating Machine), was built under the leadership of S. A. Lebedev between 1946 and 1950 in Kiev under very difficult, post-war conditions (Khomenko, 1989). In 1950, Lebedev moved to Moscow, where he continued developing digital computers, first as the head of a laboratory, and later (1953-1973) as director at the Precision Mechanics and Computer Technology Institute (ITMVT) (Bardizh, 1987). The machines developed between 1950 and 1965—the BESM (Large Electronic Calculating Machine), BESM-2, -3, -4, and -6—were all world-class machines at the time they were built, both in performance and in levels of innovation. The BESM-6, a one-MIPS (million instructions per second) machine in production between 1965 and 1980, was the workhorse of Soviet scientific computing for more than two decades, and has logged perhaps more hours of use per unit manufactured than any other computer in history.

The Soviet Union had a long history of research in parallel computing as well. Some of the earliest Soviet research was carried out by E. V. Yevreynov at the Mathematics Institute of the Siberian Branch of the USSR Academy of Sciences (Yevreynov et al., 1962, 1966). Over a period of two decades, Yevreynov experimented with the development of several "homogeneous computer systems" based on a number of hardware platforms. Fundamentally, such systems consisted of a multiple number of computational modules that could be as simple as a processor with memory or as complex as a complete computer configuration with I/O (input/ output) devices, external memory, etc. During the 1960s, a number of other Soviet researchers such as V. A. Torgashev, A. V. Kalyayev, V. V. Ignatushchenko, and V. Ye. Kotov began developing ideas that would later form the core of important parallel computer development projects funded during the late 1970s and 1980s. In the mid- to late-1960s, S. A. Lebedev also began working on parallel computing projects at ITMVT.

Between roughly 1965 and 1980, very few new high-performance computers were introduced in the Soviet Union. While research and development of high-performance and parallel systems took place at ITMVT, the Control Problems Institute (Moscow), the Cybernetics Institute (Kiev), the Computer Center and the Mathematics Institute of the Siberian Branch of the USSR Academy of Sciences (Novosibirsk),

the Leningrad Aviation Instrument Building Institute, the Multiprocessor Computer Systems Scientific Research Institute (Taganrog), and elsewhere, only ITMVT introduced a viable machine, the AS-6. This uniprocessor, real-time system was used for mission control during the Apollo-Soyuz spaceflight.

This decade and a half was chiefly characterized by the building of a very extensive Soviet computer industry as part of large-scale international programs to develop the Unified System (ES) series of mainframe computers and small system (SM) series of minicomputers. The ES mainframes were based on IBM's System/360, /370 computers, and the SM machines were based on minicomputers developed by Hewlett-Packard and Digital Equipment Corporation.[1]

As shown in Figure II.1, Soviet projects involving the development of concrete parallel machines proliferated starting in the late 1970s.[2] This reflected both a maturing of research projects to the point where the ideas could be implemented and growing needs within the military, space, geophysics, and aviation communities for powerful computers. The lag between Soviet and Western computing capabilities was also growing rapidly. During the 1970s, several new high-performance machines had been introduced in the West by Control Data Corporation, and Cray Research, Inc., founded in 1973, had revolutionized supercomputing with its vector-pipelined Cray computers.

---

1 N. C. Davis and S. E. Goodman, "The Soviet Block's Unified System of Computers," *Computing Surveys* (ACM), *10*, 2(Jun 1978), 93-122.

C. Hammer, A. G. Dale, M. B. Feldman, S. E. Goodman, W. K. McHenry, J. Schwartz, S. T. Walker, and S. Winograd, *Soviet Computer Science Research*, McLean, Virginia: Foreign Applied Sciences Assessment Center/Science Applications International Corporation, 31 Jul 1984.

2 P. Wolcott and S. E. Goodman, "High-Speed Computers of the Soviet Union," *Computer* (IEEE), *21*, 9(1988), 32-41.

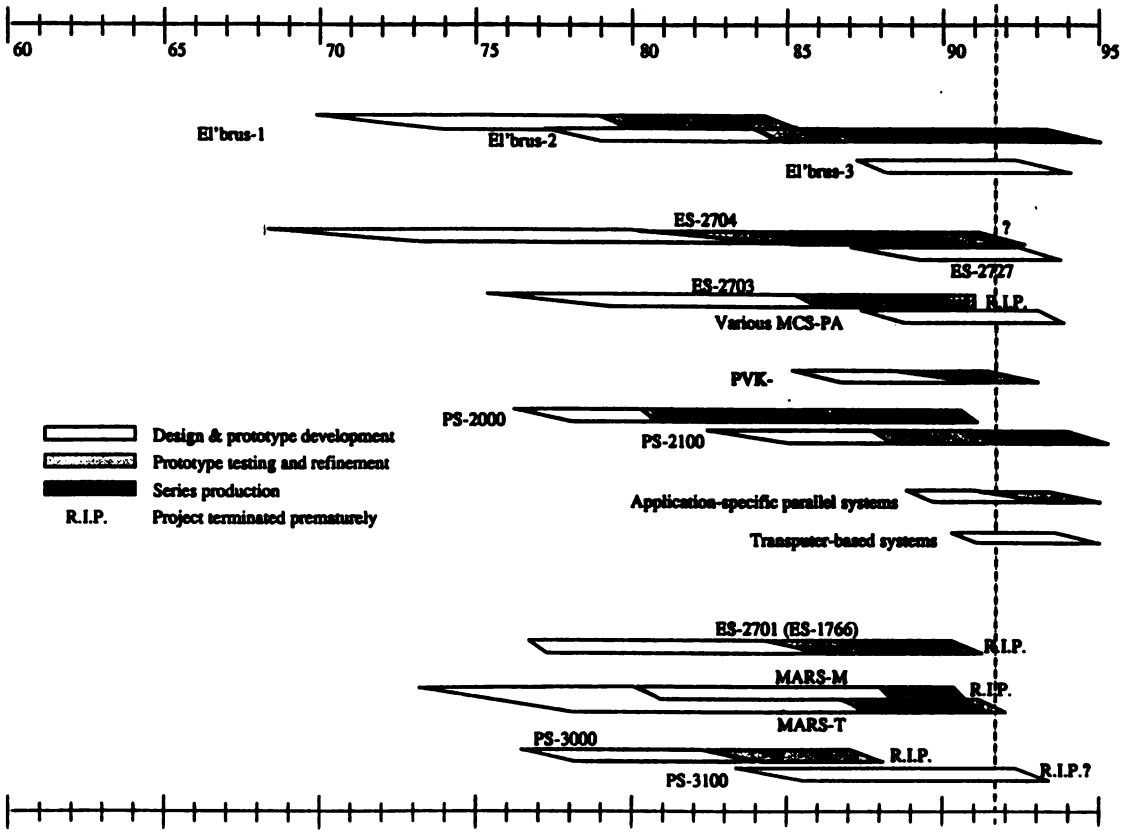


Figure II.1
Soviet Parallel Computers

The Electronic Computing Technology Scientific Research Center (NITsEVT) in Moscow, the lead R&D facility for the ES mainframes, played an important role. While NITsEVT did not develop parallel machines, it did support a number of novel architecture projects. Starting around 1980, NITsEVT provided funding and technical assistance to projects at the Cybernetics Institute (Kiev), the Multiprocessor Computer Systems Scientific Research Institute (Taganrog), the Informatics and Automation Institute (Leningrad), and others. Research projects had been in progress at these institutes for a number of years, but they were focused on developing high-performance computational elements rather than complete computer systems. In particular, I/O facilities were very weak. NITsEVT offered to provide financing and standard ES components and subsystems to help build working prototypes. The ES-2701, ES-2703, and ES-2704 were all implemented as special processors attached

to ES mainframes that provided I/O capabilities. They all incorporated ES power supplies, cabinets, cabling, etc.

Together, the parallel projects undertaken by the Soviet Union showed respectable breadth. As can be seen from Table II.1, Soviet machines reflect most of the basic categories of parallel system development found in the West. However, parallel hardware research showed little depth. There were fewer groups building parallel machines in the Soviet Union than in the West, so there are fewer groups pursuing similar ideas. Those groups that are active have a desire to differentiate themselves from other groups both in the East and in the West. One exception is the El'brus VLIW work, which is being carried out at three institutes: the Precision Mechanics and Computer Technology Institute (Moscow), the Mathematical Machines Scientific Research Institute (Yerevan), and the Computing Systems Scientific Research Institute (Moscow).

During the late 1980s, some new projects were initiated, but a number of projects were terminated for lack of funding. NITsEVT had committed to funding the ES-270x machines through to the prototype stage, but the developers themselves had to find industrial customers if their machines were to achieve series production. Convincing industry to purchase the rather exotic ES-270x machines would have been difficult under normal circumstances, but the *perestroyka* reforms compounded the difficulties. Gaining more control over and responsibility for their own finances, organizations were reluctant to spend money on unproven technology. Budget cuts reduced the amount of money available for sponsoring projects.

This chapter analyzes parallel hardware projects that are still viable. These are the El'brus, PS- , dynamic architecture machines, and the multiprocessor computing systems with programmable architecture. Although the ES-2704, one of the dynamic architecture machine prototypes, has not entered series production, we suspect that development of a successor is continuing. The ES-2703, a programmable architecture machine project, has also been terminated, but several other, related, hardware development projects continue at the Multiprocessor Computer Systems Scientific Research Institute. The hardware portions of other projects such as the ES-2701 and the MARS machines have been terminated, although some of the software and theoretical work continues. These projects are discussed in Chapter III.

Table II.1
ARCHITECTURAL APPROACHES OF SOVIET PARALLEL PROCESSORS

Machine	Maximum No. Processors	Architecture
ES-2704	24 computational modules 12 switching modules 6 interface modules	Reduction/Dataflow. "Recursive architecture." Distributed memory, multistage, switched interconnections.
ES-2727	27 computational modules 36 switching modules	
ES-2701	192 arithmetic 32 control 32 switching	Macro-pipeline. Distributed memory, multistage, switched interconnections
ES-2703	64 macroprocessors	Programmable architecture large-grain dataflow. Shared and distributed memory.
PVK-020, -460, -1600	4, 64, 256 macroprocessor signal processors	
El'brus-1, -2	10	Large-grain parallelism. Stack-based architecture, shared memory.
El'brus-3	16	VLIW processors with seven functional units. Local and shared memory.
Mini-El'brus	8	Multiple VLIW processors.
Supermini-El'brus	8	Multiple VLIW processors.
MARS-T		Network of 32-bit processors with interprocessor communications primitives in instruction set. Some transputer-like ideas.
MARS-M	4	Asynchronously communicating subsystems with multiple functional units.
PS-3x00	4 scalar 2 vector	Vector "pipeline" processors with shared memory.
PS-2000	64	SIMD architecture with flexible interconnections between processors.
PS-2100	640	Multi-SIMD architecture.

Focusing on systems with true parallelism, this chapter does not discuss high-performance systems based on the traditional vector-pipelined approach. Machines in this category include the Elektronika-SSBIS ("Red Cray") developed at the Cybernetics Problems Institute (Moscow), the ES 1191 mainframe with vector processors developed at the Electronic Computing Technology Scientific Research Center (Moscow), and the Modular Pipeline Processor (MKP) developed at the Precision Mechanics and Computer Technology Institute (Moscow). Systems with a number of attached array processors, such as the ES-1068.17, are also not examined.

C. DISCUSSION OF SOVIET WORK

1. El'brus

a. Introduction

In 1970, researchers led by Vsevolod Sergeyevich Burtsev and Boris Artashesovich Babayan began work on a multiprocessor computer that they named the El'brus, after the highest mountain in the Caucasus. The El'brus-1 and -2 were developed between 1970 and 1985; the El'brus-3 is currently under development. In developing these machines, the Precision Mechanics and Computer Technology Institute (ITMVT) played a dominant role in pushing forward the level of development in the Soviet computer industry. The El'brus machines stretched the limits of Soviet capabilities on all sides, from architecture to power supplies to cooling systems to cables to microelectronics and computer-aided design systems. ITMVT played a leading role in the development of all parts of the El'brus systems and in bringing the Soviet computer industry up to the necessary technological levels.

The experience of some Western machines such as the ILLIAC IV show that the development of a machine that pushes all the technological boundaries at once is a slow and painful process, even under the best of circumstances. Fifteen years elapsed before series production of an El'brus machine (the El'brus-2) occurred, even though billions of rubles were spent, and nearly all of the customers for these machines were in the military-industrial complex.

ITMVT continues to define the leading edge of high-performance computer development in in the successor states of the former Soviet Union in the area of

coarse-grain parallel systems with few processors. Although many El'brus ideas parallel the work of Burroughs in the case of the El'brus-1 and -2, and Multiflow Computer, Inc.,and Cydrome, Inc. in the case of the El'brus-3, it would be incorrect to say that the El'brus work copied these Western efforts. More accurately, having embryonic ideas of their own in these directions, Soviet researchers were inspired by the Western developments, and through a study of the strengths and shortcomings developed projects that were in many ways improvements of the Western work, at least at the design level. Many of the ideas incorporated into the El'brus computers were, at the time they were conceived, novel and largely unproven throughout the world. That they appeared less innovative by the time they were implemented is more a result of long development cycles than a lack of creativity on the part of the designers.

b. El'brus-1, -2

The El'brus multiprocessor design was conceived in answer to two basic needs. First, even during the mid 1960s, the components being produced by the Ministry of the Electronics Industry (*Minelektronprom*) were not very reliable, especially those produced at the limits of the production technology. A machine would have to be designed to compensate for this unreliability. Second, the primary applications for which the El'brus was designed—real-time control of objects such as spacecraft and missiles—required high reliability and programmability. The design that ultimately evolved from these requirements included up to 10 processors that were linked to a shared pool of eight memory modules, each containing four banks, and up to four I/O processors. The El'brus-2 has a clock cycle of 47 nanoseconds (ns), and up to 144 Mb of interleaved shared memory. The El'brus architecture has the following basic characteristics (Babayan et al.,1990):

- Modularity. The basic systems—central processing units, memory, and I/O systems—were modular in nature. During run-time defective modules could be shut down or swapped out and their jobs transferred to other modules without needing to shut down the entire system.

- Hardware Control. Hardware support for the detection and correction of all single errors, and detection of double errors was incorporated. The redistribution of jobs was handled by the operating system, but the hardware pro-

vided for much low-level error detection and correction without the involvement of the operating system.

- Hardware Support for High-Level Language Constructs. A desire to improve programmability caused designers to examine ways to support high-level languages more directly. This support is provided primarily through the use of a stacked-based architecture, and hardware tags. A stack-based architecture simplifies the storage of data related to procedures. Procedure calls and returns, context switching and context protection are provided in hardware.

- Tagged Architecture. While the El'brus machines operate on 64-bit data, a word is 72 bits long. The eight non-data bits form a hardware tag that permits data typing at the hardware level. In the El'brus, it is possible to represent in hardware several classes of data. These include arithmetic-logic data formats such as real and whole numbers of various lengths, arrays, and bit sets; formats that indicate addresses of data (descriptors, names) and control constructs (procedure markers and return addresses); control formats used by the operating system for working with code in physical memory; and specialized data formats such as semaphores.

  By using such tags, the hardware can distinguish between data formats and tune functional units accordingly. This helps reduce the number of essential components required, since a single functional unit can be made more versatile. Since the hardware can distinguish between data types, addresses, descriptors, and so forth, it can provide context protection at a low level, and a degree of semantic control over computation.

  The use of tags, a stack-based architecture, and other mechanisms enabled the El'brus developers to implement high-level and interpreted languages effectively (Safonov, 1988). The basic El'brus language, the procedural high-level language El'-76, was implemented efficiently enough that it could be used for systems programming as well as applications. Unlike most Western operating systems, the El'brus operating system contains no routines written for efficiency's sake in an assembler-level language. This design decision proved very fortuitous in the later development of the El'brus series.

- Dynamic Instruction Scheduling. In keeping with the orientation towards hardware control, the El'brus hardware schedules instruction execution in the multiple functional units at run-time. Instructions could be scheduled "out of order," provided the operands were available and no data dependencies were violated. A similar approach to scheduling has, in the West, recently become widespread in "superscalar" processors commonly used in workstations.

None of the El'brus-1, and -2 functional units are pipelined. According to developers, they did not sufficiently appreciate the importance of pipelined processing when designing the machine during the early and mid-1970s.

The El'brus-1, and -2 share much of the high-level design philosophy of the Burroughs 700 series machines that were introduced during the 1960s and 1970s. This includes a modular multiprocessor with shared-memory memory, a stack-based architecture, and hardware tags. The implementation details vary considerably. According to developers, a Burroughs machine was delivered to the Soviet oil industry in 1977 and ITMVT specialists were given the opportunity to examine it in great detail. The Burroughs machines helped them understand what problems to avoid, so in many respects the final El'brus design is an improvement on the Burroughs. For example, the designer of the El'brus telecommunications processor spent a half a year studying the disassembled code for the processor, then developed a high-level language for telecommunications that was more universal, and more efficient than the Burroughs language.

The El'brus machines, unlike the Burroughs models, incorporated cache memory in each processor. The El'brus solution to problems of cache coherence were rather advanced when first implemented. The hardware tags make it possible to distinguish between data types stored in cache memory. Consequently, the El'brus-2 cache is divided up into four sections: instructions, local variables, arrays, and global variables. Of these, coherence problems only arise in the case of global variables. The other categories incorporate elements that are either completely local, or used once. In the worst case, therefore, only a small portion of cache needs to be updated or erased when other processors modify memory. The result is that the El'brus-2 can effectively incorporate more processors than can other multiprocessors, such as

those manufactured by IBM. Some performance tests on the El'brus-2 show a speed-up factor of 9.85 in going from one processor to 10 (Volkonskiy et al., 1989).[3] This application, which was carefully coded to match the El'brus-2 architecture, involved the computation of the motion of an ideal gas in domains with complex geometry, using a finite-difference method of approximating the solution of two-dimensional equations of adiabatic gas dynamics in LaGrange variables.

To improve reliability, the El'brus-2 incorporates an automatic reconfiguration system. This system includes specialized hardware distributed among the processors, systems busses, and operating system facilities to automatically reassign a process from a failed module to a working one. When an error or failure in a module occurs, the hardware detects the failure, informs other modules, detaches the failed module from the configuration, and determines whether to resubmit the failed process, or invoke the operating system. The operating system processes diagnostics information and tries to restore the module to a working state when possible. It is important to note that the detection of errors, reassignment of processes, and detachment of failed modules takes place automatically. The process continues to execute even while technicians may be fixing a detached module. The systems operator can manually detach portions of the configuration. Such facilities are not found on many leading general-purpose high-performance machines in the West.

The automatic reconfiguration system integrated well with the process scheduling system. The operating system maintains a queue of processes that are ready to execute. As processors become available, they begin execution of the next process in the queue. If a processor fails, its process is placed back in the queue, and re-executed when a working processor becomes available.

The hardware error detection and automatic reconfiguration facilities not withstanding, the El'brus-2 is considered rather unreliable by users. A principal difficulty was the mismatch between the complexity of the components and the technological capability of the Soviet microelectronics industry. The efforts to place so much run-time control in hardware made the components, both logic and memory, very complex. Until recently, the Soviet microelectronics industry has not been able to manu-

---

3 Admittedly, this involved some reprogramming to tune the program for a multiprocessor configuration.

facture reliable components of this complexity. Nearly all El'brus-2 processors incorporate ECL chips mounted six to a single substrate. The mean-time-to-failure of such processors by some estimates was on the order of 92 hours (Burtsev, 1987). More recently, the contents of a single substrate have been incorporated into a single ECL gate array, reportedly significantly improving reliability in the newer models.

Between 150 and 200 El'brus-2 processors (probably on the order of 50 installations) have been manufactured to date. Production is likely to continue at least through 1992.

Because of the lack of pipelining and the limitations encountered in dynamic scheduling, each El'brus-2 processor has a theoretical peak performance of 9.4 MFLOPS, several times lower than the theoretical peak performance of 42.4 MFLOPS for a vector-pipelined processor with a comparable (47 ns) clock time, producing two results per cycle.

c. El'brus-3

In 1987, a state order for a machine with a theoretical peak performance of 10 GFLOPS was given to ITMVT. At that time, Babayan and others stopped work on the El'brus-2, and began developing the El'brus-3.

Early on, designers realized that they would have to develop a significantly new design. There were a number of basic problems. First, the functional units were not pipelined, greatly reducing performance *vis-a-vis* vector-pipelined machines with a comparable cycle time. Second, much information about instruction and data dependencies that can be determined from the source code was not available to the dynamic scheduler at run-time. The scheduler could look ahead up to 30 instructions to identify instruction and data dependencies, but this was frequently insufficient, especially in the case of conditional control transfers. Third, the dynamic scheduling made diagnostics very difficult. It was impossible to determine statically the exact order in which instructions executed. Furthermore, the changes in scheduling from one run to another produced performance figures that varied from one run to another. It was uncomfortable for developers not to be able to replicate claimed performance when demonstrating the machine to high-level officials!

To gain more control over execution and use more of the data and instruction dependency information contained in a program, developers began considering the possibility of giving the translator knowledge of, and control over, each execution cycle. In other words, the compiler would have sufficient knowledge of instruction execution times, memory access times, and transmission delays that it could schedule execution at a very fine-grain level. Developers initially were not sure that such a machine could be built, but were encouraged by the experience of Floating-Point Systems, Inc. (FPS), which has built array processors in which programmers are given very low-level control over the scheduling of each functional unit. The problem with the FPS products is that programming at such a low level is very complex; in practice a small number of FPS programmers write routines that execute directly on the array processors. These are incorporated into libraries that are accessed by applications programmers.

Babayan and his co-workers decided to combine the best parts of the FPS approach and the El'brus-2. From FPS, they took the idea of static, program-control over each cycle. The compiler, rather than the programmer, was given this control, however. From the El'brus-2, they kept the ideas of modularity, support for high-level languages, protection schemes, multiprocessing with low-overhead cache coherence mechanisms, multiple functional units and independent compilation of procedures. The earlier decision to provide enough hardware support for high-level languages that highly efficient systems software routines could be programmed in them proved very fortuitous. Because there was no assembly-level El'brus-2 code, compatibility need only be maintained at the level of El'-76, the native language of previous members of the El'brus series. Therefore, the El'brus-3 could have an entirely different architecture, yet still execute the estimated one million lines of systems and application software developed for the El'brus-1 and -2.

The architecture they developed uses what is now known as a very-long-instruction-word (VLIW) approach. This is one approach to instruction-level parallel (ILP) processing.[4] Other approaches to ILP are superscalar and data flow. The three vary in terms of how dependencies between instructions are specified and which part of the system (programmer or compiler vs. hardware) specifies dependencies and

---

4 Joseph A. Fisher and B. Ramakrishna Rau, "Instruction-Level Parallel Processing," *Science, 253,* 5025(13 Sept 1991), 1233-1241.

makes scheduling decisions. VLIW places the burden of specifying dependencies and making scheduling decisions on the compiler. In VLIW terminology, basic units of computation such as addition, memory load, and branch are called operations. These correspond to instructions in traditional sequential architectures. A VLIW instruction is the set of operations that are to be executed simultaneously. The compiler schedules a program by forming instructions out of operations that can be executed simultaneously.

In the West, two lines of VLIW research culminated in the development of commercial machines: the Trace family developed at Multiflow Computer, Inc., under the leadership of Joseph A. Fisher; and the Cydra-5 developed at Cydrome, Inc., by B. Ramakrishna Rau and others.[5, 6, 7] The El'brus-3, developed independently of either of these machines, differs significantly from them in architecture and construction. Nevertheless, there are a great many similarities in approach, especially between the El'brus-3 and the Cydra-5. Multiflow and Cydrome have both gone out of business, and their research teams have been scattered across Silicon Valley, where many team members are now working on superscalar ideas. The El'brus-3, if it enters series production on schedule, will perhaps represent the leading commercial VLIW machine in the world. Currently at the stage of prototype development, the El'brus-3 is already at or near the state of the art of VLIW research.

By 1987, largely through the design efforts and persistence of ITMVT, the Soviet microelectronics industry had made sufficient progress that the development of a machine with a 10-ns cycle was conceivable. Emitter-coupled logic (ECL) gate arrays with 1,500 gates/chip and minimum and average gate delays of 400 and 800 to 900 picoseconds (ps) were becoming available to designers.[8] To reach the goal of

---

5 Robert P. Colwell, W. Eric Hall, S. Chandra Joshi, David B. Papworth, Paul K. Rodman, and James E. Tornes, James E., "Architecture and Implementation of a VLIW Supercomputer," in *Proc. Supercomputing '90*, Los Alamitos, California: IEEE Computer Society Press, 1990, 910-919.

6 Joseph A. Fisher, "The VLIW Machine: A Multiprocessor for Compiling Scientific Code," *Computer (IEEE), 17*, 7(1984), 45-53.

7 B. Ramakrishna Rau, David W. L. Yen, Wei Yen, and Ross A. Towle, "The Cydra 5 Departmental Supercomputer Design Philosophies, Decisions, and Trade-offs," *Computer (IEEE)*, 22, 1(1989), 12-35.

8 In comparison, the early Cray X-MP (prototype in 1982) used approximately 16-gate array circuits. The Cray X-MP uses 2,500-gate array circuits, and the new C90 uses 10,000-gate array VLSI circuits. These gate arrays are all based on ECL technology.

10 GFLOPS performance, Babayan and others designed a tightly coupled multiprocessor consisting of 16 processors, each with nine pipelined functional units (five arithmetic) that could generate a result in each 10-ns cycle. Such a machine would have a theoretical peak performance of 8 GFLOPS.[9] Gate-level simulation showed that a 12.5-ns clock period was needed to allow the necessary data transfer at each stage of computation, so the theoretical peak performance of the current design is 6.4 GFLOPS. Each processor has 2 Mwords (64+8 bits) of local memory, and the full configuration includes eight 32-Mword sections of shared memory and eight I/O processors (Babayan, 1989; Dorozhevets and Wolcott, 1991).

A fundamental difference between the El'brus-3 and the Multiflow and Cydrome machines is that the El'brus-3 incorporates VLIW processors into a shared-memory, multiprocessor environment, preserving some of the strengths of the El'brus-2. The El'brus-3 makes no attempt to do a full static analysis and scheduling of an entire program, as do the Trace and Cydra-5. Not only would this be virtually impossible, given the timing variabilities and indeterminacies of a multiprocessor environment, but it would also violate some basic El'brus philosophy: the use of modularity to facilitate rapid recovery of execution in case of module failure, protection schemes, and separate compilation of procedures. Therefore, only the execution of a procedure is scheduled statically. The procedures that make up a program are assigned dynamically at run-time to available processors, as in the El'brus-2. To preserve the ability to compile procedures separately, the El'brus-3 does not use inline substitution of procedures as in the Trace and Cydra-5.

There are therefore some cases in which the static schedule of an El'brus-3 process must be altered at run-time: the overlapped execution for two independently scheduled procedures, communication with asynchronous memory, and exception handling. The first problem is unique to the El'brus-3. The unique El'brus-3 solutions in these three cases are described in (Dorozhevets and Wolcott, 1991).

The general approaches taken to loops and branches are quite similar in the El'brus-3 and the Cydra-5, although key differences exist. Both machines are ori-

---

9 In the literature, the El'brus-3 designers incorrectly include the two logic units (which do not generate floating-point results) in the calculation, giving the published figure of ~11 GFLOPS (Babayan et al., 1989).

ented toward efficient computation not only of scientific applications, like the Trace, but also of general-purpose applications. The latter often contain loops with conditionals and recurrences that are not easily handled by computers oriented towards vector processing. The solution in both the El'brus-3 and the Cydra is to create a separate context, called an iteration frame, for each loop. An iteration frame contains registers for all the variables to be accessed within a loop. Loops are therefore viewed as separate distinct computations, executing in parallel, subject only to data dependencies between them. Even when data dependencies exist, the execution of consecutive loops can generally be overlapped, improving performance. Unlike the Cydra-5, the El'brus supports look-ahead loading of data for the next iteration, and nested loops.

The Trace machines use "Trace scheduling" to execute branches. Such scheduling causes the most "likely" branch to be executed before the branch condition is known. If the scheduler guesses incorrectly, compensation code is inserted to correct the error. In contrast, the El'brus-3 provides eager execution of all branches until the branch condition is determined. Such an approach may "waste" functional units on computation that will later be discarded, but once the condition is determined, no time is wasted correcting incorrect results. The Cydra-5 also uses eager execution of all possible execution paths.

While the El'brus-3 and Cydra-5 have similar approaches to loops and branches, the latter had little influence on the former. The El'brus-3 design was rather well formulated when developers became aware of the Cydra-5 in 1989. In any case, they had no detailed information about the Cydrome machine.

d. Future

The next two to three years will be devoted to completing the El'brus-3.[10] Several El'brus-3 prototypes are currently under development, and it is expected that they will be completed during 1992. Because the components, subsystems,

---

10 Note added in proof: At the end of February 1992, Sun Microsystems announced a contract with the El'brus advanced development team to support software development work of interest to Sun by about 50 members of the team in Moscow. It is not yet clear how Sun will try to alter the software research program at ITMVT. This agreement does not seem to provide support for further El'brus-3 hardware development.

architecture, software, and production technology for the El'brus-3 are being developed simultaneously, much time will be spent debugging and tuning the machine before series production can be achieved.

A key question will be the effectiveness of the compiler. One of the reasons Western VLIW machines were not as successful as had been hoped was that "magic compilers," capable of correctly and efficiently scheduling operations on such machines, were very difficult to build. Although Soviet researchers claim that results so far show great promise, this will have to be verified through actual use when the compilers are completed.

In the West, there has been a partial movement away from VLIW to superscalar approaches, reflected in the research activities of former Multiflow and Cydrome employees. The El'brus line has moved in the opposite direction, from the superscalar-like El'brus-2 to the VLIW El'brus-3. The future direction is not clear. Having deliberately moved away from the superscalar-like ideas, it is unlikely that Babayan will move towards them again unless the VLIW ideas encounter fundamental problems. It is also difficult to predict the future evolution of the El'brus-3. Western VLIW research does not point to a clear path beyond what the ITMVT engineers have achieved. Nevertheless, regardless of what the individual processors look like, it is likely that the next El'brus will involve multiple, large-grain processors with shared memory executing El'-76.

2. Reconfigurable System (PS) Multiprocessors

a. Introduction

Since the mid-1970s, researchers at the Scientific Production Association (NPO) "Impul's" have been developing high-performance parallel computing systems for use primarily in geophysics applications. Computer designers here and at the Control Problems Institute in Moscow at this time were inspired by work on the ILLIAC IV parallel processor and Goodyear's orthogonal computer STARAN. The ILLIAC IV, developed at the University of Illinois and at Burroughs Corporation during the late 1960s and early 1970s consisted, in its final version, of a single array of 64 processing elements executing one instruction stream. The STARAN, completed by Goodyear in 1972, was a bit-oriented computer with "orthogonal organization" in

which data could be accessed not by words, but by bit-slices across many words. These ideas are incorporated in associative, or content-addressable memory concepts. The Soviet researchers were attracted by the prospects of using parallelism to overcome the deficiencies of the Soviet component base to achieve high performance.

Between 1976 and 1978, researchers designed a single-instruction, multiple data stream (SIMD) computer consisting of 64 processors with a theoretical peak performance of 200 million arithmetic operations per second on 24-bit data. The primary customers for the machine were, and to this day remain, geophysicists. The primary applications include data processing for seismic prospecting, the processing of radar and satellite images of the earth's surface, the processing of X-ray and ultrasound tomograms. The machines are also used in process control, and in real-time modeling of various physical processes (PS-2100 Computer, 1989).

During the design phase, designers rejected the idea of an associative processing field simply because an appropriate component base was not available at that time. In the interests of developing the system as fast and as cheaply as possible, developers decided to use only components that were then in series production by the Soviet microelectronics industry.

Prototypes of the PS-2000 were constructed during 1978 and 1980, and, in 1981, the machine went into series production. The rapid development and introduction of the machine into series production (by Soviet standards) is attributed to the use of the existing component base, the presence of experienced hardware designers, and an orientation towards industry. The latter served as a type of "Occam's razor" that favored simple, implementable design decisions in place of intellectually interesting, but more complex designs.

Two hundred units of 8 to 64 processing elements were manufactured before series production ended in 1990. The extensive user base has shaped, and will continue to shape, the evolution of the PS-2x00 line. The need for compatibility with existing machines precludes the introduction of radically new architectural ideas, even those, like the associative processor, which had been considered earlier.

b. PS-2100

The successor to the PS-2000, the PS-2100, is a multi-SIMD machine which, in maximum configuration, consists of 10 "base modules," each containing 64 processing elements. Each base module operates in a single-instruction multiple-data (SIMD) fashion, but different modules can process independent instruction streams. Like many parallel machines in the West, the PS-2000 and PS-2100 incorporate a general-purpose host computer, here called a monitor system, for software development, and program initiation and management. Parallel routines, stored either in a program library or loaded from the monitor system, execute on the parallel processor.

The basic architecture of the PS-2100 parallel processor and the processing elements are similar to that of the PS-2000. The PS-2x00 machines do not offer novel approaches to parallel processing, but the design decisions reflect the desire to build a usable machine. The PS-2x00 are a unique combination of various parallel processing ideas that are known in the West. One unusual element is the system of interconnects between the processing elements. While SIMD machines in the West typically use nearest neighbor, hypercube, or multistage broadcast interconnect schemes, the PS-2x00 machines use three types of connections between processing elements. A parallel bus links each processing element with two neighbors (nearest-neighbor interconnects in Western machines typically link each processing element with four or eight neighbors); a serial bus links all processing elements, but can be segmented to cluster them in groups of eight, 16, 32, or 64, so that data local to a few processors can easily be shared. A third channel allows direct memory exchange between the control unit, the external memory system, and the processing element memories. The decision to link only the two nearest neighbors via parallel links was made primarily because other parallel links would have required too much hardware and space in the PS-2000. An analysis of the algorithms to be run on the machine showed that such links plus a segmentable serial link would support their execution adequately. Over the years this decision has been reexamined, but each time the need to maintain compatibility and the hardware costs of alternative solutions cause developers to return to the original solution.

The main body of a program runs on the monitor processor, with the parallel routines off-loaded onto the parallel processor where they execute under the control

of the control unit. The off-loading can be done at run time, or prior to run-time in which case the routines are initiated by executing the appropriate call command.

The primary goals for the PS-2100 were to (i) raise the overall performance by an order of magnitude; (ii) incorporate the new component base that had become available; and (iii) overcome the chief weaknesses of the PS-2000. The latter included the short word-length, long memory access times within each processing element, the small volume of external memory, and low data transfer rates between external memory and the processing elements. The overall processing rate was increased primarily by enabling up to ten 64-processing-element base modules to be combined into one system. The appearance of CMOS gate arrays played a very significant role. The new components base allowed an eight-fold reduction in the size of each processing element. While eight boards were used for each processing element in the PS-2000, in the PS-2100 processing elements based on CMOS (complementary metal-oxide semiconductor) gate arrays with 400 gates per chip occupy only one board. The processing rate of each set of 64-processing elements is 150 MIPS, actually lower than the PS-2000, albeit with a greater word length. The theoretical peak performance of the full configuration is 1.5 GIPS.

The word-length was raised from 24-bits to 32, offering the 64-bit computation (double precision) that is a standard in high-performance machines in the West. Processing elements have eight times more local memory (128 to 512 kb) than in the PS-2000, and a vastly expanded register file for fast storage of temporary variables.

The PS-2100 offers a vastly improved external storage system, in principle, for three reasons: it is possible to attach more disks, each disk can have a higher capacity and data transfer rate, and a separate bank of semiconductor memory is included in the system. Each base module has a total of eight I/O channels with a combined peak throughput of 48 Mb/s. While not high when measured against the external storage devices used by Western high-performance computers, this is more than the 25 Mb/s of the PS-2000 (PS-2100 Computer, 1989),[11] and sufficient for the disks available in the Soviet Union. A disk subsystem, of which there can be several, has in the past consisted of up to six 317.5-Mb disks with 1.19 Mb/s data transfer rates. In part because of disrupted trade relationships with Bulgarian disk manufacturers and

11 Prangishvili (1989), a secondary source, puts this figure at 16 Mb/s.

the reluctance of Soviet disk manufacturers to produce such disks, they are in very short supply. At NPO Impul's, the majority of PS-2100 units are being equipped with hard disks from personal computers.

The solid-state semiconductor memory used as a pseudo-disk represents the first time we have seen the use of this kind of storage in Soviet machines. Large amounts of solid-state memory have been used for many years in Western supercomputers to provide the high throughput rates needed to support high-speed calculations. There can be two 16-Mb and one 64-Mb units. Each of these has a transmission rate of 5 to 8 Mb/s through a port to the systems switching device.

With a 140-ns clock time, each PS-2100 base module has a theoretical peak processing rate of 150 MIPS on 32-bit fixed-point data, 65.3 MFLOPS on 32-bit floating-point data, and 7.6 MFLOPS on 64-bit floating-point data. The theoretical peak performance of a full, 10-base module configuration is therefore 1.5 GIPS (32-bit fixed-point data), and 76 MFLOPS (64-bit floating-point data).

The PS-2100, in contrast to many other Soviet parallel projects, is an available, viable, and rather reliable machine. Like its predecessor, the PS-2000, its architecture does not reflect radical or necessarily novel approaches, but reflects common sense and a desire to produce a truly workable machine. By using standardized interfaces between base modules, the control system, and to peripherals, the PS-2100 can easily be configured to match user requirements. To provide even more flexibility in adapting to user requirements, efforts are underway to allow additional machines—from a PC to the IBM RS/6000 workstation—to serve as hosts.

The PS-2x00 machines reflect a compromise between problem-oriented and general-purpose computers. Like other machines with SIMD architectures, the PS-2100 clearly works best on applications exhibiting high degrees of data parallelism. The use of multiple types of communications channels between processors and the ability to "turn on" and "turn off" individual processors during the computation process provides a flexible system of inter-processor data exchange and computation that allows the PS-2100 to operate efficiently on a wide range of data-parallel problems.

While a vast improvement over the PS-2000, the PS-2100 suffers from numerous performance bottlenecks. The available component base has not allowed developers

at NPO Impul's to build a fast scalar processor that could process the serial portions of code. Thus the system can easily fall victim to Amdahl's Law. This law states that the achievable performance of a computer that has a parallel component and a scalar component is a function of the speed of the scalar unit and the fraction of the computation performed on it, and that the degree of speedup is dominated by the scalar portion. Specifically, Amdahl's law states that if R is the ratio of the parallel processing rate to scalar processing rate, then the final speedup, S, of a process that is P percent vectorized is

$$S = \frac{1}{([1-P] + P/R)} .$$

In the PS-2100, scalar computation can be performed in one of two ways. The host computer can execute the source program, with only the parallel routines being off-loaded onto the parallel processor. Thus the host computer can execute the non-parallel portions of the code. On a PS-1001 host, a 32-bit, fixed-point addition requires 980 ns (1.02 MIPS) (PS-1001 Control Computer, 1989). The execution time on a single PS-2100 processing unit is 420 ns (2.38 MIPS). Thus the ratio R of parallel execution rates to scalar is (64 x 2.38)/1.02 = 149. If a process is 90-percent parallelized, the total speed-up would be only 9.4, or 6.3 percent of the theoretical maximum speed-up of 149, which would occur it the process were 100-percent parallelizable.

An alternative is to perform scalar computation on a single processing element. In this case, the ratio R = 64, and the total speed-up for the example above would be 8.8, or 13.8 percent of the maximum speed-up of 64.

The PS-2100 also does not perform well on floating-point computations. The lack of high-speed floating-point units such as those produced by Weitek has forced the Impul's developers to design their own floating point units in their gate arrays. While a fixed-point add requires three clock cycles, a double-precision floating-point addition requires 60 (PS-2100 Computer, 1989). A price is paid for multiplication as well. Multiplication operations can take from 1.8 to 6 times longer to execute, depending on whether the operands are 32- or 64-bit, and fixed- or floating-point.

These statements are confirmed by benchmark tests run during September, 1990.[12] The tests also indicate that the PS-2100 performs poorly on scatter-gather operations (indirect memory access in which the addresses are stored in an array) and inner-product operations. The latter is surprising, since inner-production operations parallelize nicely. The results may have been a fluke, but are not encouraging if true, since inner-product computation plays a very important role in a wide range of scientific applications.

It should be noted, however, that the PS-2x00 can perform remarkably well on particular problems, carefully programmed to match the machine's architecture. In September 1990, a group from Sandia National Laboratories visited the Applied Physics Institute in Novosibirsk, which was solving high-speed projectile impact problems on a dual PS-2000 configuration. The PS-2000, probably tuned for such an application, produced a result in roughly the time required by a single-processor Cray X-MP, although at a courser grain (see Chapter IV).

The PS-2100, like many parallel machines, operates on the host-parallel processor principle, in which the user interfaces with a host running familiar systems software. Programming is typically done in ASPS, a FORTRAN-like language that includes low-level constructs for manipulating registers within processing elements. Programming is not easy, but similar statements can be made about most parallel machines found in the West.

The PS-2x00 machines are closest in spirit to machines like the MP-1, recently introduced by MasPar Computer Corporation. The MP-1 is a SIMD machine available in configurations containing 1K to 16K processing elements, running at 1,600 to 26,000 MIPS (32-bit fixed-point addition) and 36 to 580 MFLOPS (64-bit floating-point multiply/add average). Like the PS-2100, this system is characterized by design simplicity, CMOS technology, conservative clock rates (70 ns), and volume component replication, making it a high-performance, low-cost machine.

---

12 Peter Wolcott, *Soviet and Eastern European Computer Peformance: Results of Benchmark Tests*, MOSAIC Technical Report No. 1991-003-I, Tucson: University of Arizona, 15 Mar 1991.

c. Future

Future developments are constrained by the need for compatibility with existing machines. Fundamental changes in the basic architecture are unlikely. The changes will come from the incorporation of a new component base, and the adaptation to other host machines. In particular, developers at NPO Impul's have offered to adapt the PS-2100 to any host machine a customer might suggest, provided the customer commits to purchasing several of the machines. As a result of IBM's interest in selling machines to the Soviet oil industry, plans have been made to incorporate IBM's RS/6000 workstation as a host. If implemented, the PS-2100 would be outfitted with a fast scalar processor that would ease the pain of Amdahl's Law somewhat.

A second major change will be the use of chips based on CMOS gate arrays with, reportedly, 17,000 gates. This will make it possible to place four processing elements on a board, rather than one. In turn, this will make it possible to build a machine with more processing elements, faster execution times, and more memory on each processing element.

Impul's designers have begun talking about such a machine, called the PS-2300, which is now in the early design stages. According to early reports, each base module will contain 256 processing elements, each processing element will contain 128 kb of memory, and each processing element will execute at 4 MFLOPS for 64-bit floating-point adds, giving a theoretical peak performance of 1 GFLOPS.

The PS-2300 will incorporate a buffer-memory not found, apparently, in the PS-2100. We have no further details about it. Networking capability will be greatly expanded as well, although this is be done in the host uni-processors. Current plans are to have a prototype developed by 1994, although given the current economic crises in the former Soviet Union, this date may not be realistic. Under the new economic conditions of self-accounting *(khozraschet)* and decreased budgets, industrial enterprises are less willing and able to invest in large-scale systems. In an effort to adapt to a changing market, developers have designed a 32-processing-element version of the PS-2100. This configuration is designed to serve as an accelerator to a personal computer or engineering workstation.

It is theoretically possible that researchers in the successor states of the former Soviet Union would look to a machine like the MP-1 for some ideas for further development. For example, as they put more and more processing elements on a single board, they might be increasingly attracted to mesh interconnects between processing elements. They might consider moving to a more global interconnect like the MP-1's global multistage crossbar router network. However, the current PS-2x00 architecture seems rather well established and appears to be adequate. It will define the development path in the successor states of the former Soviet Union for at least the next generation.

3. Multiprocessor Computer Systems with Programmable Architecture

a. Introduction

The Multiprocessor Computer Systems Scientific Research Institute (NIIMVS) was established by a resolution of the RSFSR Council of Ministers in 1973 (NIIMVS, 1989a). It is located in Taganrog, a city of a few hundred thousand residents on the coast of the Sea of Azov. Founded by Anatoliy Vasilyevich Kalyayev, this organization is involved in a wide range of research and development projects in the area of multiprocessor computer systems. The triad of theory, design, and construction is applied to five principal areas of research:

- multiprocessor computer systems with programmable architecture, under Oleg Borisovich Makarevich;
- signal processing computers, under Anatoliy Ivanovich Grechishnikov;
- VLSI components with programmable structure for use in multiprocessor systems, under Aleksandr Ivanovich Khalyavko;
- neuro-computing, under Yuriy Anatol'yevich Bryukhomitskiy;
- robotics, under Igor Anatol'yevich Kalyayev.

Under the guidance of A. V. Kalyayev, work at NIIMVS has developed into one of the major Soviet lines of multiprocessor computer development. The work is orig-

inal. Although precise figures are unavailable, it appears that the Taganrog organizations collectively have focused more research time and energy on multiprocessor computer systems than have any other Soviet development groups that we know about, besides the El'brus projects at the Precision Mechanics and Computer Technology Institute in Moscow. In recent years, NIIMVS has consistently won national research funding competitions, including those held by the State Committee on Science and Technology (GKNT) and the State Committee on National Education (formerly the Ministry of Higher and Specialized Secondary Education). Since its creation, NIIMVS has sponsored a bi-annual multiprocessor computer systems conference that draws participants from a large number of organizations, both in industry and in the Academy of Sciences, even though it is sponsored by an organization in "the periphery," far away from Moscow.

b. Design Philosophy

A. V. Kalyayev is a strong personality who is able to pursue a goal single-mindedly until it is achieved. The creation of such a large and significant research center so far from Moscow is strong evidence of this. In large part because of Kalyayev's dominance over the research agenda at NIIMVS, the work here exhibits great ideological consistency. The following characteristics permeate nearly all the research at NIIMVS (Kalyayev and Nikolayev, 1988):

- Programmable Architecture. Computer performance is very much a function of how well a computer architecture is suited to a particular algorithm (or how well the algorithm can be adapted for a given architecture). Some multiprocessor architectures are fundamentally better suited for certain classes of problems than others. A machine with an architecture that closely mirrors the patterns of data flow and the granularity of parallelism inherent in the problem may have a significantly lower price/performance ratio than a machine whose architecture does not, even though the later incorporates more powerful processing elements, higher data transmission rates, and so forth.

  Kalyayev's team is well aware of this fact, and has designed a system with a "programmable architecture" that can be adapted at run-time to fit the executing algorithms. Based on a programmable cross-bar switch, the processor

field can have any configuration, which enables the formation of virtual multiprocessor systems of other classes. In particular, any standard rigid multiprocessor system can be easily programmed: bus, pipeline, matrix, hierarchical. The regularity of data flows in many algorithms makes it possible to encapsulate the necessary switching patterns in "macro-switching operations" that can be invoked during execution, reducing the programming burden for applications programmers.

- Orientation Towards "Macro-Operations." Much of the computation involved in scientific computing involves the execution of specialized algorithms: solving systems of linear equations, differential equations, Fast Fourier Transforms, calculation of eigenvalues of matrices, and many others. Many of the complex operations of such problems have been encoded into "macro-operations," which provide both an efficient implementation of these operations and give the user the ability to program using high-level mathematical constructs rather than low-level programming language constructs. Effort has been made to find the set of macro-operations that is on the one hand general enough to solve a great many numerical problems and on the other hand minimalist to keep the structure and tuning of the macroprocessor simple.

  In the general case, the macro-operations execute on so-called "macro-processors," which themselves consist of a set of computing elements that are linked by a switch, enabling direct connections between any computing elements. Thus, the programmable architecture ideas are exhibited at two levels, within a macro-processor and between macro-processors.

- Distributed Memory. Although shared memory exists, memory is also distributed among processors. "Macro-memory" components are implemented that allow complex memory access patterns to be pre-programmed, so that they can be initiated with the issuing of a single command.

- Data Flow Computation. Macro-operations are executed when the necessary input data has been received.

- On-Line Algorithms. In an effort to decrease the complexity of the switching hardware and improve performance, NIIMVS researchers have incorporated so-called "on-line algorithms" in the macroprocessors, especially in those designed for signal processing. A short introduction to on-line algorithms is given by Scott:

> In this method operands are presented serially, digit by digit, starting with the most significant, and the result digits are also formed serially at the same rate as the operand digits but with a time delay.[13]

This method requires a redundant representation to limit the propagation of carries. Having the most significant digits go first is useful in applications that involve real-time data gathering and processing, since in analog-to-digital converters the most significant digits are formed before the less significant digits.

Of the characteristics just listed, the on-line algorithms are the least prevalent, implemented predominantly in the components for signal-processing machines.

c. NIIMVS Projects

This section examines developments in three NIIMVS research projects: general-purpose computing systems with programmable architecture, special-purpose computers, and neuro-computing.

(1) General-Purpose Multiprocessors

Oleg Borisovich Makarevich has directed a number of projects to develop general-purpose computers with programmable architecture. The most prominent machine, the ES-2703, was formally initiated in 1980 as part of a government program to develop prototypes of a number of novel architecture machines. Other machines included in this program were the ES-2701 multiprocessor developed at the Glushkov Cybernetics Institute in Kiev, and the ES-2704 dynamic architecture machine developed at the Leningrad Information Science and Automation Institute.

---

13 Norman R. Scott, *Computer Number Systems and Arithmetic*, Englewood Cliffs: Prentice-Hall, Inc., 1985.

The prototype, completed in 1985, consisted of 16 processors housed in a single mainframe cabinet, and linked to an ES-1061 mainframe host. The design, however, called for up to 64 processors. The ES-2703 incorporates the notions of a programmable architecture at the inter-processor level, implemented via the programmable switching system. The macro-operations are implemented in the processors, but not in hardware, as is the case in other machines built using the microprocessors developed by NIIMVS. Rather, the ES-2703 processors were built out of Soviet analogs of the AMD 2900 bit-slice processors. By writing microcode for such chips, engineers can custom the instruction set to their purposes. The macro-operations are implemented in microcode. The processors do not support the on-line algorithms and have a more limited form of programmable memory than the home-grown microprocessors. Conventional processors were probably used because, when this research was being carried out in the early 1980s, the necessary macro-processors had not been developed to the point where they could be used in further development. By using conventional processors, researchers could concentrate on architectural, construction, and systems software issues.

The amount of main memory in each macro-processor was said to be 256 kb (Babenko et al., 1989), although this probably assumed the use of 256-kb chips that, although available now, were not available when the prototype was built. The processors do consist of multiple 16-bit arithmetic-logic units (ALUs), with a "switch" linking them. In practice, this was not a switch in the conventional sense; there is no restructuring of the links between units. Rather, microcode specified how the data were to be routed from one unit to another.

The ES-2703 did not have a very clean implementation of data flow concepts, in spite of the claims of some of the developers. While it is true that in the ES-2703 the macroprocessors do respond to commands and data as they arrive, there is much more centralized control in this system than is usual in a data flow machine. Several mechanisms can be used to manage inter-process communication: rendezvous, unconditional exchange, exchange via an asynchronous buffer, symmetrical exchange, and exchanges of messages via a "mailbox" in the core memory of the operating system (Babenko et al., 1990). The rendezvous and symmetrical exchange—where each processor sends and receives information in a loop—in par-

ticular require the explicit programming of the exchange processor, which manages the exchange in a centralized manner.

The theoretical peak performance depends on the operand length. The four 16-bit ALUs in each processor can execute four 16-bit operations simultaneously, two 32-bit, or one 64-bit. According to Kalyayev et al. (1988), the performance of a 16-processor unit is 125 MIPS on, presumably, 32-bit data. Makarevich claims the theoretical peak performance of such a unit is 64 MFLOPS (Babenko and Makarevich, 1990). Whatever the theoretical performance, a least one set of actual test data indicates that real performance is rather poor (Babenko et al., 1988). On an easily parallelized problem, the addition of two 12x12 matrices, the speedup from one to two processors was 0.26. From two to four, it was 0.0014, or virtually zero. It is not clear precisely where the bottleneck was in this test case (probably I/O), and this is but one set of results, but they do give reason to question the effectiveness of the design and/or implementation of this machine.

There are at least three projects that can be considered successors of the ES-2703. The "super macrocomputer" is a rather straightforward extension of the ES-2703 ideas. The principal differences are that it will have 96 macro-processors, an improved component base, and significantly improved I/O. I/O was a very severe bottleneck on the ES-2703. All access to external storage was made through the single, mainframe host. There was no independent access to disk storage. In the super macrocomputer, a new I/O system consisting of 32 I/O processors with direct access to disk storage will be added. The switching system will tie in directly to the switching system of the macro-processor field, giving them high-speed links to external storage. Such a configuration was not implemented in the ES-2703. Like that in the ES-2703, the super macrocomputer component base will incorporate Soviet analogs of Western chips—the Am29300/29C300 microprocessors, the successors to the Am2900. If the Soviet versions are not available, researchers intend to use original AMD products.

A second project, the "personal multiprocessor supercomputer," is a four-macro-processor system that will serve as an accelerator to a personal computer or workstation (NIIMVS, 1989c-d). It is expected to have a theoretical peak performance of 64 MFLOPS on 64-bit data. The system will include the Am29300 microprocessors.

A third project, the "personal macrocomputer," like the personal multiprocessor supercomputer, consists of four macro-processors. These units, however, will use domestically designed VLSI components that incorporate the on-line algorithms. The theoretical peak performance is expected to be 128 MFLOPS on 40-bit data. This machine is probably less general-purpose than the others, and could perhaps be one of the problem-oriented computers described in the next section.

On the one hand, the three projects reflect progress along a technological trajectory. The basic design ideology is maintained, while improvements are made in the component base, and the design is tuned to relieve bottlenecks and otherwise improve the overall parameters of the machines. On the other hand, they reflect the realities of the current economic circumstances. Designers claim that the super macrocomputer is currently only a paper design with slim prospects for implementation; there are no customers who are willing to finance the development of such a machine. The personal multiprocessor supercomputer and the personal macrocomputer reflect the need for researchers in the former Soviet Union to decrease their development cost/time and serve a larger market.

(2) Problem-Oriented Multiprocessors

Anatoliy Ivanovich Grechishnikov is the head of the division working on problem-oriented computer systems. There are applications, including signal and image processing, that require only a limited amount of functionality, but high processing rates. It is often beneficial to design a special-purpose machine and strip it of unnecessary (and costly) functionality. Grechishnikov builds systems for real-time digital signal and image processing applications: radar, sonar, geophysics, communications systems, medicine, and for natural resources research.

These systems all draw on Kalyayev's fundamental ideology of programmable architectures consisting of processors that implement complex, special-purpose functions in hardware. In many writings and lectures Kalyayev has commented on the flexible nature of his institute's architecture. The special-purpose machines are examples of how the overall ideology can spin off a number of niche-oriented machines. The architecture permits the interconnection between processors to be varied from task to task, the basic set of hardware executable functions to be altered or expanded to meet the needs of the specific application, and bit-lengths of the data to be varied,

all without major changes to the hardware. Grechishnikov has taken advantage of these features, building several machines using components developed at NIIMVS.

We are aware of three models in this family of problem-oriented computing systems (PVK): PVK-020, PVK-460, and PVK-1600. These systems have theoretical peak performances of 20, 460, and 1600 MIPS. The PVK-020 has a 16-bit wordlength, while the PVK-460 and PVK-1600 use 24-bit operands. These figures reflect the theoretical peak performance of each machine in full configuration. The machines are designed using vector, scalar, and control units that can be combined in various configurations. Thus, PVK-460 configurations, for example, can have processing rates ranging from 60 to 460 MIPS (NIIMVS, 1989e). This flexibility permits systems to be produced easily for a wide range of customer needs and budgets. According to developers, maximum configurations of all three prototypes have been assembled, and it is likely that the PVK-020 and PVK-460 are in regular, if not series, production.

(3) Neuro-Computing

The four individuals spearheading the neural networks research in Taganrog are Yuriy Anatol'yevich Bryukhomitskiy, the head of the NIIMVS neural- and robotics systems department, Gennadiy Anatol'yevich Galuyev, head of the NIIMVS neural systems laboratory, Yuriy Viktorovich Chernukhin, professor at the Taganrog Radio Engineering Institute, and Vladimir Ivanovich Bozhich.

Work on neural network systems has been carried out at NIIMVS since roughly 1975. While the Soviet Union, like the West, experienced an initial period of excitement and activity in neural network research that was followed by unmet expectations and a decrease in activity, researchers at NIIMVS continued working on these problems even when few others were. They are now, we believe, among the leading researchers in this field in the former Soviet Union.

Work of this group is distinguished from many other neural network projects in that it is a completely digital approach. There are two basic reasons for their selection of this approach. First, they view this path as the most promising for the future. In general, there are three basic approaches to neural-network development. One is to model the neural network in software on a general purpose machine. A second is to build a co-processor to a personal computer. A third, and historically the first, is

to develop a completely analog system. The first two approaches do not have the necessary performance, according to Bryukhomitskiy. While analog systems have reasonable speed, they also become less and less reliable as the number of processors increases; they don't handle changes in temperature as well as digital systems. The digital approach, on the other hand, depends on the availability of VLSI chips. Although only gate-array chips with 3,000 gates are currently available and they need chips with at least 10,000 gates, developers feel that the current trend towards greater levels of integration will eventually give them the chips they need. Furthermore, the Taganrog group has been encouraged by the opinions of R. Hecht-Neilson,[14] who attended an international conference on neural networks in Moscow in 1989, where he stated that he too feels that digital neural networks are the general course and intends to pursue this direction.

A second reason is historical. NIIMVS work grew out of the work on multiprocessor computer systems with programmable architecture being carried out in Taganrog. It was natural that they should adopt an approach based on digital parallel systems with the crossbar interconnects.

The basic element currently used was developed in 1987 and contains one neural element per chip. It is therefore called a neurochip. We have no concrete figures about the level of integration of this chip. A brochure printed in 1989 mentions two other chips that have been prototyped, or are near the prototype stage.

The first, of which prototypes exist, is a chip based on a CMOS gate array chip with 2,200 gates embedded in a standard 64-pin circuit carriers (NIIMVS, 1989b). Its threshold operating frequency is 10 Mhz and has an eight-bit storage capacity. Six of these neurochips can be integrated on a single silicon substrate. Besides the six gate arrays, this chip set includes five MSI chips that provide the interconnect logic. It is this programmable interconnect that allows connections between the neurochips to be reconfigured at will. A second chip is being designed that will be based on a CMOS gate array chip with 10,000 gates. This chip will incorporate multiple neurons, although it is not clear how many. Bryukhomitskiy mentioned that a chip that embodies 16 neurons was under development with expected completion time of

---

14 A prominent US neural network researcher; founder of HNC, Inc., a California-based neural network hardware, software, and research services company.

1991 (NIIMVS, 1989b). The designers were anticipating that the system would be capable of 10^{10} to 10^{12} connections per second.

The developers feel that their systems could be used in many applications that traditional computers handle poorly: image processing, pattern recognition, robot control, and other artificial intelligence problems. This is the standard list of applications mentioned by neural network researchers throughout the world.

Some of the work is presented by Kalyayev et al. (1989). According to Western neuro-computing experts, this work does not make any significant contributions to the field of neural networks, unless it is in the implementation. Details about the implementation are still too sparse to make definitive conclusions about this. It does not appear that this system uses any models that are not known in the West.

There are some indications that Bryukhomitskiy's projects are not proceeding as well as planned. Some unconfirmed reports state that his prototype chips are not functioning properly. In 1990, Bryukhomitskiy's division was split into two. One of Bryukhomitskiy's laboratories, the robotics laboratory and at least one of the neural networks laboratories, that headed by Vladimir Ivanovich Bozhich, has been transformed into a separate division.

As described by Kalyayev and Bozhich (1990), Bozhich's architecture is based on a hypercube topology between, theoretically, up to 2^{10} neural processors. Bozhich describes an algorithm to broadcast information from each neural processor to every other processor in the shortest amount of time. The work of Bozhich is significant in two respects. First, this is one of very few examples of the use of the hypercube interconnect topology anywhere in the Soviet Union. Second, it is the first real break from the traditional use of a full crossbar interconnect at NIIMVS. It is, perhaps, the first time that researchers here are seriously working on the question of linking together massive numbers of processors. In a hypercube configuration, if there are $m = 2^n$ processors, each processor is linked with n neighbors. As has been common knowledge for over a decade in the West, the number of links in such a topology grows only as O(log) of the number of processors.

It is likely that the Taganrog neuro-computers will be of greatest benefit, at least in the short run, as tools to support experimentation in neural network research. As

in artificial intelligence and parallel processor machine research, the ability to experiment is extremely important. The behavior of complex, indeterminant systems cannot be described in an accurate and detailed manner using theory alone. Growing out of the programmable architecture tradition, the NIIMVS neuro-computers incorporate the ability to reconfigure easily the "synapses" between the neurochips (alter the topology) and program the neurochips to implement various neural models. One can therefore use the same hardware to model and experiment with many different neural models. In particular, the current system can be used to implement the formally logical, dynamic, and gradual models that are described by Kalyayev et al. (1989). Systems that implement models in software also have such flexibility, but often at the cost of prohibitively poor performance.

d. Related Work in the West

Many of the concepts incorporated into the multiprocessor computing systems with programmable architecture (MCS PA) such as on-line algorithms and signed-bit representations, reconfigurable architectures, crossbar switching mechanisms and data flow, also have been worked on in the West. The contribution of the NIIMVS projects to the state of computing today lies not so much in the development of untried architectural approaches, but in a) providing a unique combination of architectural features in a single system, and b) adding data about the functioning of specific architectural features—individually and when combined into a system—to the world knowledge pool. It does not advance the state of the art, but is academically interesting.

There may be specific low-level solutions to the design of the hardware and software of the MCS PA machines that could contribute in real terms to high-performance computer development today. Further research needs to be done to find these solutions, or demonstrate that they don't exist.

(1) On-Line Algorithms

The on-line methods were introduced by M. D. Ercegovac, who developed on-line algorithms for multiplication, square roots, and division.[15, 16] Stanishevskiy and others acknowledge the Western origin of these ideas, but claim that they have developed additional algorithms. Scott mentions that

> . . . application of this technique to floating-point arithmetic has also been studied, although floating-point is complicated by the separate treatment needed for exponents and for significands.[17]

NIIMVS researchers have apparently implemented such algorithms.

In implementing these algorithms, NIIMVS researchers have used a "signed-bit binary representation" (Kalyayev and Nikolayev, 1988). Although we do not know exactly what it is, it is likely one of the "redundant signed-digit" representations discussed by Avizienis (1961). This type of representation can provide operations in which carry propagation is absent. Although it requires more circuitry, this system has potential for high-speed operation.[16]

(2) Architectures

Problems with parallel processing systems with fixed interconnection topologies have been recognized by many. Kalyayev points primarily to the fact that the performance of a system is heavily dependent on the "fit" between the application and the underlying architecture that executes it (Kalyayev, 1986; Kalyayev et al., 1988; Kalyayev and Nikolayev, 1988). A number of Western projects have incorporated reconfigurable architecture ideas. Three of these are the Remps system developed by Kai Hwang at the University of Southern California, the Kyushu University Reconfigurable Parallel Processor, and the CHiPs system developed by Lawrence Snyder at Purdue University.

---

15 K. S. Trivedi and M. D. Ercegovac, "On-Line algorithms for Division and Multiplication," *IEEE Trans. Computers, C-26,* 7(1977), 402-406.

16 V. G. Oklobdzija and M. D. Ercegovac, "An On-Line Square Root Algorithm," *IEEE Trans. Computers, C-31,* 1(1982), 70-75.

17 Norman R. Scott, *Computer Number Systems and Arithmetic,* Englewood Cliffs: Prentice-Hall, Inc., 1985.

The Remps system is a MIMD computer that consists of N identical processors, each of which consists of M pipelined processing elements.[18, 19] Remps uses a number of different interconnection systems. Within each processor, processing elements are connected to each other by a routing network called a pipeline net. The processors are linked with each other through an interprocessor network. The processors are linked with shared memory blocks through a third interconnection system, a crossbar "global network."

Among Western projects, the Remps machine is perhaps the most like the MCS PA. Like the MCS PA, it incorporates parallelism at two levels, between processors and within processors, and uses a macro-dataflow computational model. Remps has a more expanded and flexible shared memory system than does the MCS PA. Data for external devices is placed in the shared memory, rather than being linked directly to the processors, as is planned for future MCS PA systems. Both systems provide shared memory as well as memory local to each processor.

A main difference is the nature of computation that is executed at the different levels. While in the MCS PA each processor is assigned the execution of a macro-operation, each Remps processor is assigned a task, generally (although not always) a higher-level construct. Thus the discussion of reconfiguring the architecture to linear, tree, matrix, or hexagonal topologies in the MCS PA takes place at the level of inter-macroprocessor connections; in Remps, the pipeline net linking processing elements provides these topologies. The MCS PA does add an additional concept, the group, to support the execution of a task. Like the MCS PA macroswitch that links macroprocessors, the Remps pipeline net is configured using explicit commands. While the MCS PA uses a set of synchronization constructs, the Remps, at the processing element level, relies on programmed delays through the crossbar interconnect to synchronize operations.

---

18 K. Hwang, "Multiprocessor Supercomputers for Scientific/Engineering Applications," *Computer (IEEE)*, (Jun 1985), 57-73.

19 K. Hwang, Zh. Xu, and A. Louri, "Remps: An Electro-Optical Supercomputer for Parallel Solution of PDE Problems," in *ISC '87. 2nd Int'l. Conf. on Supercomputing, Supercomputing '87, 1* (1987), 301-310.

The Remps does not appear to incorporate the complex memory management chips (macromemory) that are being developed for the MCS PA.

In summary, the Remps project takes a very similar approach to the MCS PA to solve a very similar set of applications. The Taganrog work clearly preceded Hwang's, however, and has been implemented to a much fuller degree. Hwang was not aware of the Taganrog work until recently. No performance figures are available for the Remps since it has not been implemented. Therefore no substantial quantitative or qualitative discussion of performance is possible.

The Kyushu University reconfigurable parallel processor is an MIMD multiprocessor with N processing elements fully connected by S NxN crossbar networks.[20] The system that was scheduled to be completed at the end of 1989 consists of 128 processing elements and a single 128x128 crossbar switch. Memory is completely distributed, although processors are able to access portions of memory physically located at another processor, enabling the system to be configured either as a distributed memory system, or as a shared memory system.

This system differs from the MCS PA in a number of significant ways. It lacks the multilevel parallelism, relying on conventional microprocessors that execute conventional instruction sets. It does mirror the MCS PA approach to building an interconnect system. The 128x128 crossbar switch is composed of a composite of smaller switches. The basic modules are 8x8 LSI crossbar switch chips. A 16x16 matrix of 256 such chips are combined to create the full switch. The 8x8 switches use 1.2 micron CMOS technology, run at 20 MHz, and have 16 switching patterns. The approach at NIIMVS has been to create single-chip 8x8 or 16x16 switches as the basic building blocks. When switches of higher order are needed, multiple chips are combined. The Kyushu machine demonstrates that this approach is a viable one, even when only moderate levels of integration are used. The 1.2 micron level is still beyond the capabilities of NIIMVS, but is probably within reach of the microelectronics industry.

---

20 Kazuaki Murakami, Akira Fukuda, Toshinori Sueyoshi, and Shinji Tomita, "An Overview of the Kyushu University Reconfigurable Parallel Processor," *Computer Architecure News, 16,* 4(Sept 1988), 130-137.

The Configurable, Highly Parallel (CHiP) computer was developed at Purdue University by Lawrence Snyder and others during the 1980s.[21] It is a family of architectures that share three basic components in their construction: homogeneous microprocessors, a switch lattice, and a controller. The processors are distributed in a two-dimensional grid. The switches that constitute the lattice are located between the processors at regular intervals. The setting of each switch in the lattice is governed by configurations that are stored in memory local to each switch. The controller loads the configurations and is responsible for broadcasting the commands that cause the switches to tune to a particular setting. CHiP computers can differ from each other in the number of switches between processors, the number of processors, the number of data paths connected to a processor, and so forth.

While the CHiP computer differs greatly from the MCS PA, it represents another school of reconfigurable architecture machines that has been pursued in the West.

(3) Systems with Macro-Operations

In nearly all discussions of the MCS PA, designers comment on the fact that complex algorithms are implemented in hardware, or in microcode, resulting in a faster implementation and enabling the user to program at a higher level than offered by conventional programming languages. In the West, the implementation of such algorithms has been chiefly in the form of libraries of specialized software routines that are called from within applications. From the point of view of the scientific applications programmer, the calling of routines from a standard library is no more cumbersome than invoking a macro-operation command on an MCS PA.

A powerful trend in the development of microprocessors has been the Reduced Instruction Set Computers (RISC). During the 1970s, researchers at IBM found that about 20 percent of the instruction set commands were being executed 80 percent of the time. This discovery led to efforts to implement a fewer number of instructions in a more efficient and effective manner. The resulting RISC approach is characterized by fewer instructions, fewer transistors needed for fabrication, making it possible to implement the processor on a single chip, and a minimal number of cycles per

---

21 L. Snyder, "Introduction to the Configurable Highly Parallel Computer," *Computer (IEEE), 15,* 1(Jan., 1982), 47-56.

instruction and faster clocks because of the streamlined, on-chip operation of the processor. In short, RISC is the antithesis of the macro-operation approach proposed by Kalyayev.

Although the issue of whether RISC or Complex Instruction Set Computers (CISC) is better has not been settled, many of the most powerful microprocessors in existence are RISC, and the approach is now widely accepted as viable. At the same time, RISC vendors have added helpful CISC features such as floating-point hardware, memory management units, cache memories, pipeline control circuitry, and hardware multipliers. CISC manufacturers are beginning to incorporate reduced instruction set approaches as well. Thus the gap is narrowing (Inmos, 1989).

Although levels of integration are increasing, it is not very likely that Western chip manufacturers will move towards a macroprocessor type system. When improvements in manufacturing technology allow more transistors to be placed on a single chip, the general preference is to use that space to provide more memory rather than more complex functionality for general-purpose machines. When specialized functions are implemented in hardware, it is generally for application-specific integrated circuits in which the performance payoff is very high. Such chips are typically not programmed as are general-purpose microprocessors.

Processors like the Am29C327 floating-point processor do support the microprogramming of complex functions. In addition to being microprogrammable, it has an eight-deep register file for storing immediate results used in recursive operations, and facilitates compound operations such as Newton-Raphson division and sum-of-products through the 64-bit datapath.

e. Research Support at NIIMVS

There are two basic sources of funding for NIIMVS research: contracts and the state budget. The research directions clearly reflect the nature of these sources. In the past, industrial organizations were allocated funding that had to be spent on research. Much of this money was spent with little accountability, and the final products were often simply paper designs or basic mock-ups. As a result of the system of self-accounting *(khozraschet)*, sponsors are paying for research using "their own" money, and are much more concerned that they obtain usable results.

A related effect, seen particularly in the case of the personal macrocomputer and the personal multiprocessor supercomputer, is the trend towards attempting to develop smaller-scale, niche products that are more affordable, more easily developed, and more in line with customer requirements. The problem-oriented machines are also designed so that they can easily be modified to suit the needs of customers. This trend is, to a degree, reinforced by the fact that sponsors are now allocating money for only a year at a time, rather than the three to five years that was the norm earlier.

Most of the financing through the state budget is for fundamental research. Here an opposite trend can be seen. Two major programs—"Supermacrocomputer" and "Promising Information Technologies," funded by the State Committee on National Education and the State Committee on Science and Technology—are supporting the development of large-scale projects that involve a combination of neural networks, macro-processor based systems, and component research. Projects on each of these topics were selected from the 20 to 30 proposals submitted to the State Committee on Science and Technology, two of them taking first and second place.[22] According to NIIMVS personnel, NIIMVS has won such GKNT competitions for three years in a row.

f. Future Directions

The future track of NIIMVS research is likely to be very strongly guided by the existing ideology of multiprocessor computer systems with programmable architecture, at least as long as A. V. Kalyayev remains director of the institute. In a talk given in 1991, Kalyayev reconfirmed the basic design philosophy behind his machines.

Future development of applied research projects is likely to consist of incremental improvement of the existing design approach. Such developments will include improving the throughput of I/O systems, increasing the performance parameters of the components, and enhancing the systems software to use the underlying hardware more effectively.

---

22 We are not certain which two took the top places.

It is more difficult to predict the future of the fundamental research. First, we have little information about these projects. One developer claims that the goal is the creation of a real-time supercomputer that integrates aspects of artificial intelligence, knowledge bases, and neural-computing into a single, unified system. This is an ambitious goal at best. Second, there is some evidence that research in this area is deviating from some points of the traditional ideology. The ambitious goals may force researchers to modify some of their traditional approaches. The use of a hypercube topology in Bozhich's neural networks research has been mentioned. There are also reports of research projects working on software for imported transputers. NIIMVS researchers increasingly have been incorporating transputer terminology into their talks. The macro-processor has even been termed a "super-transputer." It is not clear that such talk implies that future work will incorporate such transputer features as the communications protocols or the NEWS (north-east-west-south) nearest-neighbor topology.

Ultimately, the progress that can be made in building future machines is limited by the available component base. The chip manufacturing equipment currently installed at NIIMVS can fabricate chips with 1,000 gates. With some in-house tuning of the equipment, chips with 3,000 gates may be possible, but this is the limit. If researchers had access to chips with 5,000 gates, however, they feel they could implement an entire macroprocessor on a single chip, significantly improving performance and reliability. Of course, higher levels of integration would be even more desirable. The microelectronics industry of the successor states of the former Soviet Union is probably capable of producing such chips. If NIIMVS gets access to such technology—domestic or foreign—significant improvements in their products will become possible.

4. Dynamic Architecture Machines

a. Introduction

Valeriy A. Torgashev and others have conducted research on dynamic architecture machines (MDA) since the late 1960s. Earlier references to these machines used the term "recursive architecture" machines. The earliest presentation of some embryonic ideas is found in work by Glushkov et al. (1974). Arguing that the von

Neumann architecture—characterized by low-level machine language representations, sequential execution, and linear storage in memory—required a burdensome amount of complex software to bridge the distance between the hardware and the applications to be executed, the authors proposed a so-called recursive architecture solution. Such a machine would be characterized by a recursive internal language, a recursive parallel method of control, a recursive memory organization, and recursive internal and external architectures. The internal language would allow a programmer to define a set of low level primitives and then construct additional levels of language elements that themselves would be viewed as entities, but which would be decomposed during execution. Control would be based on dataflow principles in which operators (at any level) would execute as soon as all the necessary operands were available. Memory would be stored and referenced in terms of objects, rather than memory locations; the logical-to-physical mapping would be done at run-time. The internal architecture, the set of connections between processors, would dynamically change in response to the structure of the task being executed. Finally, the machine would consist of a hierarchy of physical switches that would allow one to cluster processors physically together. These ideas, expressed in very general terms by Glushkov et al. (1974) have, to varying degrees, been incorporated in Torgashev's later work on dynamic architecture machines.

Torgashev's work falls into the general category of non-von Neumann architectures that are classified as data-driven and demand-driven computer architectures.[23] In data-flow machines, the availability of operands triggers the execution of the operation to be performed. In demand-driven (reduction) computers, the requirement for a result triggers the operation that will generate it. Two primary motivations behind these developments are the desire to improve performance by uncovering and utilizing the greatest amount of parallelism in a program possible, and to support declarative languages, particularly functional languages, which have some desirable properties from the point of view of programmability and execution.[24, 25]

---

23 Philip C. Treleaven, David R. Brownbridge, and Richard P. Hopkins, "Data-Driven and Demand-Driven Computer Architecture,""*Computing Surveys (ACM), 14*, 1(Mar 1982), 93-143.

24 Paul Hudak, "Conception, Evolution, and Application of Functional Programming Languages," *Computing Surveys (ACM), 21*, 3(Sept 1989), 359-411.

25 Steven R. Vegdahl, "A Survey of Proposed Architectures for the Execution of Functional Languages," *IEEE Trans. Computers, C-33*, 12(1984), 1050-1071.

As pointed out by Vegdahl (1984), the terms functional language, applicative language, data flow language, and reduction language have been used somewhat interchangeably in the literature. Such languages free the programmer from having to worry about many features of so-called imperative languages (for example, FORTRAN, Pascal, C). Imperative languages involve changing the "state" of computation (which is implicit in the values of the program counter, the values of all variables, the stack, and so forth) by "constructs," or commands in the source language. One of the key characteristics of such languages is that program variables are thought of (and implemented as) memory locations. Two separate pieces of code may reference the same variable and unless the order of interaction is carefully managed by the programmer, unintended (and incorrect) manipulation of variables can occur.

In contrast, declarative languages have no implicit state, and programming is done entirely in terms of expressions. In functional languages, the underlying model of computation is the function, in the true mathematical sense. A function is applied to inputs, and generates a result. The value of function depends only on its textual context, not on computational history. Each function contains its own state, so different functions can be executed independently of each other, without concern for the impact of one function execution on the state of another.

The hardware/software systems that employ such non-von Neumann approaches can be examined at a number of levels. Treleaven et al. provide a survey that examines the computation organization, the program organization, and machine organization of such systems.[26] A survey of proposed architectures for the execution of functional languages, compares machine designs by looking at the interconnection topologies, program and data representation, methods of driving the computation, process management, and dynamic optimization techniques.[27]

---

26 Philip C. Treleaven, David R. Brownbridge, and Richard P. Hopkins, "Data-Driven and Demand-Driven Computer Architecture," *Computing Surveys (ACM), 14*, 1(Mar 1982), 93-143.

27 Steven R. Vegdahl, "A Survey of Proposed Architectures for the Execution of Functional Languages," *IEEE Trans. Computers, C-33*, 12(1984), 1050-1071.

This section provides an overview of the features of Torgashev's dynamic architecture machines and tries to identify the primary distinguishing features. Because publications do not provide much detail on the machines and language implementations, it is difficult to see through Torgashev's terminology to make a clear comparison. The conclusions of this section are, therefore, somewhat tentative. Torgashev's machines and the RYaD programming language are described by Ponomarev et al. (1983), Torgashev (1984), Ponomarev et al. (1987), and Torgashev and Plyusin (1989).

b. Overview of Dynamic Architecture Machines

The underlying model of computation in Torgashev's machines is the dynamic automata network (DAN). In such a network, the division of a problem into algorithms and data (typical in traditional programming) is minimized. Each automaton represents an object of a given problem, which can be data, operations, relations, references (pointer-like entities), or physical machine resources. Automata can be complex, consisting of other, simpler automata. The execution of a problem, represented by a DAN, consists of applying transformations to the DAN until no further transformations are possible. The final structure represents the solution. An important point is that not only data can be transformed, but also other types of automata, such as relations and operations.

Treleaven et al. describe the program organization as the way machine code programs are represented and executed in a computer architecture. Two basic categories of mechanisms are the data mechanism that defines the way a particular argument is used by a number of instructions, and the control mechanism, which defines how one instruction causes the execution of another instruction.[28] It is not clear from the literature how arguments/results are used in the MDA. It appears that the MDA uses at least two of the three data mechanisms described by Treleaven—by value and by reference. It also appears that the MDA makes provisions for using each of the three control mechanisms described by Treleaven et al. (1982): sequential, where a single thread of control signals passes from one instruction to another; parallel, where control signals the availability of arguments and an instruction is executed when all its arguments are available (as seen in dataflow); and recursive—where

---

28 Philip C. Treleaven, David R. Brownbridge, and Richard P. Hopkins, "Data-Driven and Demand-Driven Computer Architecture,"*"Computing Surveys (ACM), 14,* 1(Mar 1982), 93-143.

control signals the need for arguments and an instruction is executed when one of the output arguments it generates is required by the invoking instruction.

The lowest level automata are represented by so-called program elements that can consist of code fragments to execute individual instructions (operator automata), to provide simple or complex memory access (data automata), to verify a relationship (relation automata), and to manage the assignment of program elements to physical resources (resource automata).

The physical machines, as represented in the ES-2704 and ES-2727, consist of two basic components: a set of computational modules, and a set of switching modules. The computational modules consist of a set of processors—administrative, execution, control, and three switching units, in the case of the ES-2704—together with shared memory. Automata exist in memory and are queued to the execution processor. If the execution processor becomes too heavily loaded, automata are queued to the switching processors that send them to other computational modules to be executed. The ES-2704 consists of 24 computational modules, 12 switching modules, and six interface modules that provide access to external host computers, which provide access to external data stores. Several prototype models have been built using Soviet analogs of the Motorola MECL chips with three to 10 simple gates. The theoretical peak performance, according to Ponomarev et al. (1987) is 100 MIPS. The ES-2727, currently under construction, consists of 72 computational modules, 36 switching-interface modules arranged in a two-tiered hierarchy. The machine is being constructed using ECL gate arrays of unknown levels of integration. The theoretical peak performance of a full configuration is said to be 750 to 1,000 MIPS (Torgashev and Plyusin, 1989).

Treleaven et al. (1982), Vegdahl (1984), and Hudak (1989) describe a number of data-driven and demand-driven computers that have been developed over the last 15 to 20 years.[29, 30, 31] Because of a lack of detail about the MDA implementations, it

---

29 Philip C. Treleaven, David R. Brownbridge, and Richard P. Hopkins, "Data-Driven and Demand-Driven Computer Architecture," *Computing Surveys (ACM), 14*, 1(Mar 1982), 93-143.

30 Steven R. Vegdahl, "A Survey of Proposed Architectures for the Execution of Functional Languages," *IEEE Trans. Computers, C-33*, 12(1984), 1050-1071.

31 Paul Hudak, "Conception, Evolution, and Application of Functional Programming Languages," *Computing Surveys (ACM), 21*, 3(Sept1989), 359-411.

is difficult to make precise comparisons with the Western machines. The literature appears to indicate the following:

- It is not at all clear that the MDA have as rigorous a theoretical foundation as the Western work, particularly that related to functional languages and machines for executing them. Hudak (1989) reviews much of the theory of functional languages and the accompanying properties. The relationship between the dynamic automata network and mathematical functions is not clear. We do not know whether the DAN has a set of mathematical properties analogous to those that make functional programming so attractive. For example, is computation in the MDA free of side effects?

- Unlike most of the Western work, which uses either control flow, data flow, or reduction for the computation model, the MDA appear to provide constructs that can be used to specify any of these modes. Although the MDA operate primarily on a data flow principle, the relations between operators and between instructions can be specified so that control flow or reduction are in effect.

- The MDA apparently place significant emphasis on performance, perhaps compromising theoretical purity. First, physical resources within the system can be explicitly managed by a programmer. Automata can be assigned to run on specified processors, or be stored in specified portions of memory. Second, the system integrates uniprocessor and multiprocessor ideas. Each computational module can manage the execution of a number of automata concurrently, so it is possible to balance the cost of communications overhead with the benefit of distributed processing. Third, it appears that in some cases operators can be sent to the nodes at which operands are located. If operands are large data objects such as arrays, the reduction in communications overhead can be considerable.

- The MDA allow the incorporation of library routines that can, apparently, be coded in languages other than RYaD. This capability is found in SISAL (Streams and Iterations in Single Assignment Language), developed at Lawrence Livermore National Laboratories. Such features compromise the "purity" of the implementation, but have the advantage of making it possi-

ble to use existing routines, and provides a migration path to a declarative programming model without a complete rewrite of existing software.

- The MDA implementation emphasizes reliability. The switching network consists of switching processors that maintain "indicator sets" that record the status of the physical resources in the system. When processors or memory fail, automata can automatically be reinitiated on operating resources without the intervention of the user.

c. Unanswered Questions

The literature does not provide answers to a number of critical questions. For example, how do the MDA avoid redundant computation? Religious adherence to the DAN computational model could cause certain computations to be performed multiple times. Most data- and demand-drive architectures have some provision for reducing the amount of redundant computation. How do the MDA handle storage management? How do the MDA avoid deadlock?

5. Transputers

In 1990, on the initiative of Vladimir Konstantinovich Levin, the Soviet Transputer Association (STA) was formed. The name of the association is almost a misnomer, since it includes not only organizations and individuals using Western transputers, but also Soviet organizations that for many years have been developing multiprocessor systems with homogeneous processors. There have been half a dozen major projects of this nature in the Soviet Union. According to Levin, these include the work on dynamic architecture machines of V. A. Torgashev, the macro-pipeline work of I. N. Molchanov, A. A. Letichevskiy, and others at the Glushkov Cybernetics Institute of the Ukrainian Academy of Sciences, the programmable architecture machines at the Multiprocessor Computer Systems Scientific Research Institute, the Kronos projects under V. Ye. Kotov in Novosibirsk, work by N. N. Mirenkov at the Mathematics Institute in Novosibirsk, and work at the Semiconductor Electronics Institute under Khoroshevskiy, also in Novosibirsk. In Moscow, individuals at the Scientific Research Center "Kvant," and the Informatics Problems Institute are also heavily involved with transputers and transputer-like systems.

While traditionally the groups mentioned above have designed their own processors on the technological base available in the Soviet Union, nearly all of them reportedly are seeking to implement some of their machine design and software ideas on transputer platforms. Most of the above groups have several tens of real transputers with which they are experimenting. In spite of export control restrictions, TRAM units (transputers plus associated memory) are available for rubles for approximately the same amount as a PC/AT personal computer. Other groups are involved in development as well. For example, at the International Center for Informatics and Electronics (InterEVM), researchers are building multiprocessor systems consisting entirely of Western processors. These include a system consisting of four i860 processors, a four-processor system using the R3000 RISC processor, and some accelerator boards for a PC that incorporate transputers. Unlike transputer-like systems, the multiprocessors use bus-interconnections. These systems do not necessarily offer new scientific ideas, but they do provide examples of the penetration of powerful Western components into the former Soviet Union.

One of the primary motivations for forming the association was the desire to participate in the development going on in the West on such systems. An association could organize conferences, disseminate information about transputer and transputer-like systems, and serve to facilitate contact between eastern and western practitioners. Members of the association have attended Western transputer conferences where they have given papers. The association cannot finance development work, however, although members of the associate do help evaluate research proposals for such organizations as the State Committee on Science and Technology and the Academy of Sciences.

The association reportedly has on the order of 200 member organizations. Some, like the Applied Mathematics Institute (IPM) in Moscow, have long been involved in complex problems, such as aerodynamic computations. At a recent transputer conference in Glasgow, representatives from IPM reportedly gave a paper on their use of an eight-processor transputer configuration to solve some fluid dynamics problems significantly faster than on the most powerful ES mainframes and El'brus computers in operation at IPM.

At least two transputer conferences have been held in the Soviet Union: in June 1990 and October 1991.

The introduction of transputers, which are specifically designed to function in multiprocessor configurations, could have a profound influence on Soviet parallel processing research. They can relieve developers of the burden of a weak microelectronics industry, providing a reasonably powerful platform on which to develop and test systems and applications software. They can provide a standardized platform enabling researchers to share results more easily. Furthermore, since the hardware platforms will be available to users as well, the results of the research can quickly be transferred to users. This scenario contrasts sharply with the past, in which many of the results of multiprocessor research were never utilized by users, in part because the machines never made it into series production.

6. Special-Purpose High-Performance Computers

Today's supercomputers, costing millions of dollars, are cost effective in large part because they are general-purpose, able to solve problems in a wide range of applications domains. It is possible, however, to design machines that cost significantly less, yet out-perform conventional supercomputers on a very narrow set of problems.[32] For example, Norman Christ at Columbia University has built a machine to solve one problem—simulating how quarks bind to form neutrons and protons. His machine, costing a fraction of a Cray, can perform 6.5 billion operations per second. Other special-purpose machines include the Digital Orrery developed at the Massachusetts Institute of Technology by Dr. Gerald Sussman, which simulates the orbits of the planets, the GRAPE (Gravity Pipe) designed by researchers at the University of Tokyo to simulate clusters of stars or galaxies, and the Fast Data Finder, designed by TRW, Inc., to search for words in a large textual database. By tailoring the computer to a specific task, it is possible to reduce the amount of hardware, and optimize that which remains to the task at hand.

Since 1989, Soviet researchers have published a handful of designs or descriptions of application-specific processors and multiprocessors. The most fully developed is a processor built at the Landau Theoretical Physics Institute that is tailored to the Monte Carlo method for the Ising model with random links on a lattice containing 256x256 spins (Talapov et al., 1990a). This machine, the prototype of which

---

32 Andrew Pollack, "The Custom-Made Supercomputer," *New York Times* (27 May 1990), F4.

was completed in December 1989 (Talapov et al., 1990b), is capable of executing $4x10^6$ elementary Monte Carlo steps per second. Upgraded versions of this machine, using a Western component base, are reportedly under development in the West.

Vyzhikovskiy and Kanevskiy of the Kiev Polytechnic Institute have published designs for a systolic specialized processor for solving systems of linear equations using the Gaussian method (Vyzhikovskiy and Kanevskiy, 1990). Specialists at Moscow State University have designed a specialized processor for magneto-hydrodynamics problems capable of running at 1 GFLOPS (Bakhmurov and Smelyanskiy, 1989). Although the article makes it clear that such a machine has not yet been built, it presents an analysis of the requirements, showing that it could be built using currently available CMOS components. Donskov et al., at the High-Energy Physics Institute in Protvino, describe a special-purpose processor for particle sampling by momentum in experiments studying central hadron collisions (Donskov et al., 1990).

D. KEY SOVIET RESEARCH PERSONNEL AND FACILITIES

Table II.2 presents selected key Soviet research personnel and facilities working in the area of parallel hardware.

Table II.2
KEY SOVIET RESEARCH PERSONNEL AND FACILITIES—PARALLEL HARDWARE

Computing Systems Scientific Research Institute, Moscow (Russia)

Cybernetics Institute im. V. M. Glushkov, Ukrainian Academy of Sciences, Kiev (Ukraine)*

Yuliya Vladimorovna Kapitonova
Aleksandr Adol'fovich Letichevskiy
Igor Nikolayevich Molchanov

"Impul's" Scientific Production Association, Severodonetsk (Ukraine)

I'ya Israilyevich Itenberg
Aleksandr Sergeyevich Nabatov
Vladislav Vasil'yevich Razanov

Informatics and Automation Institute (Computer Center), USSR/Russian Academy of Sciences, Leningrad/St. Petersburg (Russia)

Valeriy Antonovich Torgashev

Informatics Systems Institute, Siberian Branch, USSR/Russian Academy of Sciences, Novosibirsk (Russia)*

Mikhail Nikolayevich Dorozhevets
Vadim Yevgenyevich Kotov
Aleksandr Gur'yevich Marchuk
E. V. Tarasov
Yuriy Leonidovich Vishnevskiy

International Center for Informatics and Electronics (InterEVM), Moscow (Russia)

Sergey Aleksandrovich Rakov

* Work on parallel hardware at these institutes has apparently ceased. Software development continues.

Table II.2
KEY SOVIET RESEARCH PERSONNEL AND FACILITIES— PARALLEL HARDWARE (cont'd.)

Mathematical Machines Scientific Research Institute, Yerevan (Armenia)

Mathematics Institute, Siberian Branch, USSR/Russian Academy of Sciences, Novosibirsk (Russia)*

Nikolay Nikolayevich Mirenkov

Multiprocessor Computer Systems Scientific Research Institute, Taganrog (Russia)

Lyudmila Kliment'yevna Babenko
Vladimir Ivanovich Bozhick
Yuyiy Anatol'yevich Bryukhomitskiy
Anatoliy Ivanovich Grechishnikov
Igor Il'yich Itenberg
Anatoliy Vasil'yevich Kalyayev
Oleg Borisovich Makarevich

Precision Mechanics and Computer Technology Institute im. S. A. Lebedev, USSR/Russian Academy of Sciences, Moscow (Russia)

Boris Artashesovich Babayan
Gennadiy Georgevich Ryabov
Yuliy Khapanovich Sakhin
Vladimor Yur'yevich Volkonskiy

Scientific Research Center "Kvant" *

Vladimir Konstantinovich Levin

* Work on parallel hardware at these institutes has apparently ceased. Software development continues.

(blank)

CHAPTER II: PARALLEL HARDWARE
REFERENCES

Avizienis, A., "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Trans. El. Comp., EC-10*, 3(Sept. 1961), 389-400.

Babayan (Babaian), B. A., G. G. Ryabov (Riabov), and G. D. Chinin, "Elbrus Software Methodology: Instrumentation-Experience," in *Information Processing 89, Proc. IFIP 11th World Computer Congress, San Francisco, 28 Aug 28-1 Sept 1989*, Ed. G. X. Ritter, Amsterdam: North-Holland, 1989, 879-882.

Babayan (Babaian), B. A., "Basic Results and Prospects for Development of the 'El'brus' Architecture," *Prikladnaya informatika, 15* (1989), 100-131 (in Russian).

Babayan (Babaian), Boris Artashesovich, Aleksey Vasil'yevich Bocharov, Vladimir Sergeyevich Volin, Sergey Semenovich Gavrilov, Anatoliy Sergeyevich Groshev, Fedor Anatol'yevich Gruzdov, Mikhail Viktorovich Yeremin (Eremin), Sergey Mikhaylovich Zotov, Arnol'd Leonovich Plotkin, Leonid Yevgen'yevich Pshenichnikov, Gennadiy Georgiyevich Ryabov (Riabov), Mikhail Leonidovich Chudakov, and Vladimir Stepanovich Shevyakov (Sheviakov), *Multiprocessor Computers and the Methods of Their Design*, Ed. Yu. Smirnov, Moscow: Vysshaya shkola, 1990.

Babenko, L. K., O. B. Makarevich, and L. N. Matveyeva (Matveeva), *Principles of the Organization of Computation in Multiprocessor Systems for Real-Time Applications*, Preprint 16-88, IPPMM, AN UkSSR, L'vov, 1988.

Babenko, L. K., O. B. Makarevich, A. I. Sukhinov, and O. I. Ovcharenko, "Means of Implementing Basic Computational Algebra Operations in Multiprocessor Computer Systems," *Upravlyayushchiye sistemy i mashiny* (Kiev), 5(1989), 25-28 (in Russian).

Babenko, L. K., O. B. Makarevich, and L. N. Matveyeva (Matveeva), "Mechanisms of Interaction and Synchronization of Parallel Processes in a Multiprocessor System," *Autom. Control Comput. Sci.* (Riga), 1(1990), 55-63 (in Russian).

Bakhmurov, A. G., and R. L. Smelyanskiy (Smelianskii), "The Design of an Architecture of a Specialized High-performance Computing System," in *Hardware-Software Systems and Systems Software of Computing Systems*, Moscow: Moscow State University, 1989, 19-28.

Bardizh, V. V., *Eighty-five Years After the Birth of S. A. Lebedev*, Moscow: Precision Mechanics and Computer Technology Inst., 1987.

Burtsev, V. S., "Reliability Characteristics of Multiprocessor Systems and an Analysis of the Reliability of the El'brus-2 MVK," Preprint 169, Moscow: USSR Academy of Sciences, 1987.

Donskov, S. V., A. V. Inyakin, Yu. V. Mikhailov, A. V. Sin'govskiy (Singovskii), V. P. Sugonyayev (Sugoniaev), P. M. Shagin, and A. V. Shtannikov, "Special-purpose Processor for Particle Sampling by Momentum in an Experiment to Study Central Hadron Collisions," *Instrum. Exp. Tech.*, 4(1990), 96-100.

Dorozhevets, Mikhail N., and Peter Wolcott, *The El'brus-3 and MARS-M: Recent Advances in Soviet High-Performance Computing*, MOSAIC Technical Report No. 1991-017, Tucson: University of Arizona, Aug 1991.

Glushkov, V. M., V. A. Myasnikov (Miasnikov), M. B. Ignat'yev (Ignatiev, Ignat'ev), and V. A. Torgashev, "Recursive Machines and Computing Technology," in *Information Processing 74, Proc. IFIP Cong. 74*, Ed. Jack L. Rosenfeld, Amsterdam: North-Holland Publishing Company, 1974, 65-71.

Inmos, *The Transputer Handbook*, Inmos, Ltd., Oct 1989.

Kalyayev (Kaliaev), Anatoly V., "Multiprocessor Systems with a Programmable Architecture," in *Fifth Generation Computer Architectures. Proc. IFIP TC 10 Working Conf. on Fifth Generation Computer Architectures, Manchester, UK, 15-18 Jul 1985*, Ed. J. V. Woods, Amsterdam: North-Holland, Amsterdam, 1986, 291-300.

Kalyayev (Kaliaev), A. V., O. B. Makarevich, I. A. Nikolayev (Nikolaev), and L. K. Babenko, "A Supercomputer with a Programmable Architecture," in *EVT Electronic Computer Technology*, V. 2, Ed. V. V. Przhiyalkovskiy (Przhiialkovskii), Moscow: Radio i svyaz', 1988, 161-171.

Kalyayev (Kaliaev), A. V., and I. A. Nikolayev (Nikolaev), "Multiprocessor Computer Systems with Programmable Architecture," Ch. 3 in *High-performance Systems for Parallel Processing of Information*, Ed. V. V. Gritsyka, Kiev: Naukova dumka, 4 (1988), 100-148.

Kalyayev (Kaliaev), A. V., Yu. V. Chernukhin, Yu. A. Bryukhomitskiy (Briukhomitskii), and G. A. Galuyev (Galuev), "The Concept of Digital Neurocomputers With Programmable Architecture," NIIMVS working paper, 1989

Kalyayev, A. V., and V. I. Bozhich, "Neurocomputers with Programmable Architecture," in *Multiprocessor Computing Structures: A Collection of Scientific Papers*, V. 12, Eds. A. V. Kalyayev (Kaliaev), A. N. Melikhov, and V. O. Bronzov, Taganrog Radio-Engineering Inst., XXI(1990), 4-9.

Khomenko, L. G., "History of the Creation in the Academy of Sciences Uk SSR of the First Indigeneous EVM," *Ocherki istorii yestyestvoznaniya i tekhniki, 36*, (1989), 74-81 (in Russian).

NIIMVS, "The Scientific-Research Institute for Multiprocessor Computing Systems (SRI MCS)," Brochure, Multiprocessor Computer Systems SRI, Taganrog, 1989a.

NIIMVS, "Large-Scale Integrated Circuit of the Digital Neuronlike Processor," Brochure, Multiprocessor Computer Systems SRI., Taganrog, 1989b.

NIIMVS, "Personal Multiprocessing Supercomputer (PMS)," Brochure, Multiprocessor Computer Systems SRI, Taganrog, 1989c.

NIMVS, "Personal Multiprocessor Supercomputer," Brochure, Multiprocessor Computer Systems SRI, Taganrog, 1989d.

NIIMVS, "The PVK-460 Problem-Oriented Computer System," Brochure, Multiprocessor Computer Systems SRI, Taganrog, 1989e.

Ponomarev, V. M., V. U. Plyasnin (Pliasnin), and V. A. Torgashev, *Distributed Computing and Machines with Dynamic Architecture*, Leningrad: Leningrad Research Computing Center, 1983.

Ponomarev, V. M., V. A. Torgashev, and V. D. Didenko, "Distributed Computational Systems with Dynamic Architecture in Integrated Production Systems," in *Integrated Production Systems: A Collection of Articles*, Eds. V. M. Ponomarev, V. V. Aleksandrov, A. A. Barilov, I. P. Belyakov (Beliakov) et al., Leningrad: Mashinostroyeniye, 1987, 27-34.

Prangishvili, I. V., "High-Performance Computing Systems with Reconfigurable Structure," in *Electronic Computer Technology*, Vol. 3, Ed. V. V. Przhiyalkovskiy (Przhiialkovskii), Radio i svyaz', 1989, 28-39.

PS-2100 Computer Complexes, Prospectus, 1989.

PS-1001 Control Computer System, Prospectus, 1989.

Safonov, V. O., "Concerning the Effective Implementation of Expandable Dynamic Languages for the El'brus Multiprocessor Computing Complex," *Programm. Comp. Soft.*, 4(Jul-Aug 1988), 61-67.

Talapov, A. L., V. B. Andreychenko (Andreichenko), V. S. Dotsenko, and L. N. Shchur, "Dedicated Processor for Studying Ising Model on Random Lattice," *JETP Lett.*, *51*, 3(1990a), 161-163.

Talapov, A. L., V. B. Andreychenko (Andreichenko), V. S. Dotsenko, and L. N. Shchur, *Ising Random Lattice Special Purpose Processor*, Preprint CP/89-01, Theoretical Physics Institute im. L. D. Landau, Jan 1990b.

Torgashev, V. A., "RYaD - A Programming Language for Distributed Computing," Preprint 28, Leningrad Research Computing Center, 1984.

Torgashev, V. A., and V. U. Plyusin (Pliusin), "Processors with Dynamic Architecture," in *Electronization of the National Economy*, Ed. N. V. Gorshkov, Moscow: State Committee on Computer Technology and Informatics, 1-2 (1989), 27-36.

Volkonskiy (Volkonskii), V. Yu., Ye. V. Krivosinnaya (Krivosinnaia), and Ye. N. Chernis, "Measuring the Performance of the MVK El'brus-2 on Test and Practical Problems," *Programm. Comp. Soft.*, 6(1989), 60-69.

Vyzhikovskiy (Vyzhikovskii), R., and Yu. S. Kanevskiy (Kanevskii), "The Design of Systolic Processors for a Numerically Stable Solution for Systems of Linear Equations Using the Gaussian Method," *Elektronnoye modelirovaniye* (Kiev), *12*, 2(1990), 29-36 (in Russian).

Yevreynov (Evreinov), E. V., and Yu. G. Kosarev, *On the Possibility of Constructing High-Productivity Computer Systems*, Novosibirsk: SO AN SSSR, 1962.

Yevreynov (Evreinov), E. V., and Yu. G. Kosarev, *Homogeneous High-Productivity Computer Systems*, Novosibirsk: Nauka, 1966.

(blank)

CHAPTER III
PARALLEL PROGRAMMING

A. SUMMARY

Research in parallel programming in the Soviet Union generally treated the same topics as in the West. These include automatic vectorization/parallelization translators that attempt to discover parallelism in a program with little or no assistance from the programmer, as well as the development of diverse types of programming languages that either attempt to make the parallel programming task easier or attempt to efficiently use a parallel architecture, or both.

In addition to observing that the approaches being pursued have been similar, two other generalizations are possible about the entire area. First, Soviet research in parallel programming has simply not been as abundant, in that fewer scientists have been working on the problem. Second, there has been a noticeable lack of Soviet experimental data and experience, due certainly to the limited availability of parallel computers.

On the topic of automatic vectorization/parallelization, the Soviet systems have been founded on Western techniques that have, in cases, been extended by Soviet methods claimed to be new. Though information in both communities is often insufficient to verify these claims, the nature of the area suggests that some of them are almost certainly true. However, like most of the work on the topic, the advances are incremental. There was no evidence of any Soviet breakthroughs. Indeed, the reported systems often have a "first generation" character to them, suggesting that there has been no pressure to develop more refined or more sophisticated techniques. This has probably been due to the lack of a substantial user community (a lack that persists in the successor states).

In the parallel language arena, it has been possible to find instances of Soviet work on most varieties of parallel languages, though the number of instances of each is fewer than in the West. The languages are generally copies, variants, or extensions (often substantial) of Western languages. In a few cases, truly unique new languages have been developed; MAYaK, Polyar, and BARS are perhaps among the most interesting. A dominant characteristic of most of the Soviet language studies is that no

implementation details or performance numbers were reported. Consequently, it is extremely difficult to assess how effective the research has been, but generally the work has been paralleled or surpassed by efforts in the West.

In computer programming languages, Soviet researchers from the very beginning emphasized theoretical foundations and formal analysis. It is not surprising then to find the designers of most languages citing the principles on which their design is based—the MAYaK inventors cited three antecedent lines of research (Glushkov et al., 1976, 1978, 1981)—or formulating much of their exposition in terms of theoretical foundations. This is not generally the case in the West, except in functional programming. It is impossible to determine whether this use of theory has been empowering or limiting, because there is so little experience with the resulting languages.

One characteristic of the former Soviet educational system that could explain this strong theoretical component is that there has been no computer software major in Soviet universities; it has been taught as a specialty in the Applied Mathematics degree programs.[1] The eminent academician A. P. Yershov (1984) cited this as a shortcoming of the educational system contributing to the "software problem."

In one programming language topic, the Soviet Union was competitive with the West. These are languages we will call here "coarse-grain composition languages." They are used to assemble modules written in different procedural languages into a parallel computation. Though languages of both communities have had the capability for heterogeneous composition for some time, its importance seems to have been recognized only in recent years. The capabilities offered by languages from both communities are comparable.

---

1 Committee to Study International Development in Computer Science and Technology, "Computing Technology in the Soviet Union and Other CMEA Countries," in *Global Trends in Computer Technology and Their Impact on Export Control*, National Research Council, Washington, DC: National Academy Press, 1988, 126-218.

Finally, earlier analyses of Soviet computing capability have speculated that parallel computation could be a domain where the Soviet Union might "leapfrog the West."[2] The rationale was based on two observations:

> First, Soviet universities enjoy a strong theoretical base in mathematics, which is a crucial requirement for parallel processing research and development. Second, and perhaps more importantly, Soviet computer hardware lags badly behind the West in many areas, including chip density and processing speeds. Parallel processing offers a possible avenue around these hardware deficiencies by linking together relatively inferior machines to create adequate hardware.[3]

This reasoning seems wrong. The first assertion, that a strong basis in theory is a "crucial requirement for parallel processing research," has not been borne out in Western experience. Mathematical theory seems not to have played either a positive or negative role in the West in terms of practical achievements. Secondly, "linking together relatively inferior machines" does not produce "adequate hardware," if adequacy means "high performance." The authors ignore the fact that another form of inferiority of Soviet computers is unreliability. It is often erroneously thought that connecting multiple copies of unreliable components together can achieve both reliability and performance. Though reliability *may* be achieved, that is, some parts *may* be functional at a given time, high performance is achieved only when all parts are functional all of the time. Thus, improved performance imposes higher, not lower, reliability requirements. The assertion that parallelism is "an avenue around hardware deficiencies" is, therefore, not supported.

B. INTRODUCTION

Parallel programing software has been an active research topic for a decade, though important antecedent work was conducted in the 1970s. The activity has been most intensive in the United States, though researchers have been nearly as active in Western Europe, and there have been notable efforts in Israel and Japan.

---

2 Richard W. Judy and Robert W. Clough, "Soviet Computing in the 1980s: A Survey of the Software and Its Applications," in *Advances in Computers*, Vol. 30, Ed. M. C. Yovits, Boston: Academic Press, Inc., 1990, 223-306.

3 Ibid., p. 230.

The research is motivated by an understanding that the programming and compilation tasks for parallel computers are fundamentally different from those of the original sequential computers. Two essential differences are programming paradigms and operational complexity. The paradigms of sequential computer programming concern specifying the sequence of steps necessary to solve a problem. Specifying these steps is required of parallel computation as well, but it is also necessary to state how these steps can be divided up among independent processors and applied to the common data without "unexpected collisions." New languages and language constructs are needed for expressing these new features. The complexity of programming, already substantial in the sequential case, increases dramatically in the parallel case as the programmer orchestrates the concurrent execution of the many processors. The exponential explosion of potential interactions qualitatively changes programming, precluding the possibility of preplanning the computation at the lowest level of detail. Complexity-reducing abstractions and methodologies are essential.

The parallel programming research area generally includes the subjects of parallel programming languages and compilation techniques suitable for parallel computers. Environments and tools to simplify the parallel programming task are often included with the language system. Such facilities might include debuggers, verification aids, and interface code for other languages.

Two divisions are used for the primary classification of the parallel programming software in this chapter: vectorizers and parallel programming languages. Vectorizers encompass most of the compiling technology studied. The parallel programming section contains subdivisions for extended languages, coarse-grain composition languages, functional languages, asynchronous process languages, and fine-grain parallelism.

C. VECTORIZERS

A "vector processor" is a complex subsystem of a computer allowing operations to be applied to vectors, sequences of numbers, rather than to single values as in the host processor. These computations are pipelined, a basic parallel processing technique that makes effective use of hardware and memory bandwidth. Vector units can be incorporated into sequential machines, or attached to the processors of parallel machines. "Vectorizers" are compilers that translate source programs into code

suitable for execution on a machine with a vector processor. The program analysis performed by vectorizers is also of interest for parallel machines without vector units because the information discovered about the program can often be used to generate efficient code for these machines. Accordingly, there has been considerable interest in vectorizers in both the United States and Soviet Union (Val'kovskiy and Lebedev, 1989).

FORTRAN is the most common source language for vectorizers. The Applied Mathematics Institute has produced a source-to-source translator for FORTRAN IV, that is, the output of the translation is also FORTRAN IV augmented with special function calls for the vector unit. This Fora-ES Converter translates DO-loops into vector instructions for the SPES special purpose vector processor of the PS-3000 (Zadykhaylo et al., 1987). The methodology is a modestly enhanced form of Western approaches. Reportedly, six of the 14 Livermore Loops, standard vectorizing benchmarks, were successfully translated and run, yielding speedups of 1.2 to 8 over the ES-1045 (Val'kovskiy and Lebedev, 1989). A similar system is the FORTRAN precompiler for the IZOT 1703.[4]

Another source-to-source translator for the PS-3000 was developed at the Control Problems Institute (Babichev et al., 1985). The source language is FORTRAN 77 extended to include explicit vector commands. These include declarations for specifying which dimension of an array is to be vectorized and a "colon" notation for specifying an index sequence within FORTRAN subscripts. This system produces parallel code by vectorizing loops containing assignment statements with index expressions. The parallelization is done by the linear transformation methods (Babichev and Lebedev, 1983; Trakhtengerts, 1987) and is claimed to produce results not achievable with the general method (Babichev et al., 1985).

A vector FORTRAN parallelizing translator was developed for the M-10 (Varchenko and Natanson, 1986). Only loops explicitly identified by the programmer are parallelized. Performance results have not been discussed in the literature, but they may be better than those for an earlier M-10 parallelizer for ALGOL-60. This transla-

---

4 Peter Wolcott and Seymour E. Goodman, *Recent Developments in High-Speed Computing in the USSR*, MOSAIC Technical Report No. MOS-1989-042, Tucson: MIS Department/University of Arizona, 1989.

tor assumed full responsibility for identifying the vector operations in the computation, focusing only on inner loops and applying very simplistic methods. Factors of improvement of 3 to 4 on "real" scientific applications have been reported when the programmer expresses the inner loop in a form the translator can handle. In other cases the 15- to 20-percent improvement has been described as "insignificant" (Val'kovskiy and Lebedev, 1989).

Another non-FORTRAN vectorizer is the Vector Pipelining Compiler, which vectorizes PL/1 for the vector pipelining computer (Ivannikov et al., 1986). Developed by the Cybernetic Problems Institute of the USSR Academy of Sciences,[5] the compiler vectorizes the array commands of the language and nonnested loops. Only loops meeting fairly stringent conditions are vectorized, suggesting that rather simple techniques are used. The vectorizing facility can be turned on and off, and the compiler can be told when to ignore dependencies that it cannot successfully analyze. Explicit vector instructions are produced. Four of the 14 Livermore Loops were vectorized producing a factor of improvement from 5 to 10 over scalar computation (Val'kovskiy and Lebedev, 1989).

Additional research studies focusing on problems of vectorization and parallelization have been conducted. Parallelizing loop nests for multiprocessor execution using the workpile paradigm has been reported (Konoshenko and Telegin, 1989).

Branch parallelization is available for FORTRAN, ALGOL-60 and MNEMOKOD programs within the MINIMAKS programming environment (Mirenkov, 1989). The program source text is compared against code fragments for algorithms with inherent parallelism. Matches are marked, analyzed further and allocated to parallel processors. Additional program improvement is available from automatic vector and matrix parallelization and interactive programming assistance for hardcoded parallelism.[6]

---

5 At the end of 1991, the USSR Academy of Sciences changed to the Russian Academy of Sciences, the name it used before June 1925.

6 William Curran and Peter Wolcott, *Parallel Programming in the Soviet Union*, MOSAIC Technical Report No. MOS-1991-004, Tucson: MIS Department/University of Arizona, 10 Apr 1991.

Overall, the Soviet vectorizing systems use methods developed in the United States, but which are typically enhanced or extended. Several operational systems have been produced, though the overall volume of work reported is substantially less than in the West. It is difficult to make performance comparisons because the hardware platforms are so difficult to calibrate, but if the numbers reported are meaningful, then they are generally in line with Western experience. The component missing from the Soviet vectorization literature is extensive, long-term experience with operational systems applied to everyday problems. This is surely due to the general unavailability of vectorized supercomputers.

D. PARALLEL LANGUAGES

Inventing parallel programming languages and notations has been a popular activity in the 1980s throughout the parallel computation research community. Most of these languages are not fully implemented, and those that are implemented have but few users, often only the designers. Consequently, few parallel languages have a significant impact on the state of knowledge.

1. Extended Languages

One class of parallel languages includes those that extend an existing sequential language. Because the base language semantics must be substantially preserved, few sophisticated constructs can be included. Simple structures like *parbegin, parend,* which delimit the beginning and ending of parallel activity, are typical of extended languages. Since such languages are straightforward to design and are often particularized to the features of a single machine, it is sufficient to add representative instances of non-FORTRAN extended languages to the list cited under the vectorizers section (III.C).

Pascal 2703, developed by the Multiprocessor Computer Systems Scientific Research Institute in Taganrog, translates Pascal with task grouping and synchronization operations into PA, an intermediate object-oriented parallel assembler language for the 2703 (Mirenkov, 1989); the implementation runs on an ES-1061 and a prototype 2703.[7] A parallel COBOL extension, called P-KOBOL, is under develop-

---

7 Ibid.

ment in the Siberian Branch of the USSR/Russian Academy of Sciences (Mirenkov, 1989). PARIS is a novel paper study proposing extensions to the set language SETL for parallel execution (Korneyev et al., 1990).

2. Coarse-Grain Composition Languages

An important type of parallel programming language in which the Soviet Union has demonstrated particular success does not have a widely accepted name, but can be called coarse-grain composition languages. (Examples of this type of language developed in the United States would include Poker,[8] its descendants and PCN.[9]) The principal goal of these languages is to provide mechanisms for assembling coarse-grain processes programmed in other languages to execute concurrently. Two instances of these are MAYaK and PARUS.

MAYaK, a Russian acronyin for "macropipelined language," has been developed by the Cybernetics Institute of the Ukrainian Academy of Sciences (Gorokhovskiy et al., 1984) for the ES-2701 macropipeline system. The designers assert that the language is founded on principles from Soviet theoretical work: the theory of data structures (Glushkov et al., 1976), theory discrete system design (Glushkov et al., 1978) and the theory of macropipelining (Glushkov et al., 1981).

The MAYaK language has three parts: MPP, mnemonic for multimodular programming, and two subsets, Prostor and YaDRO (Mikhalevich et al., 1986). Other subsets of the language, identified in the early language description (Gorokhovskiy et al., 1984), are no longer mentioned.

YaDRO is a low-level application programmer's language similar to standard programming languages. The primitives available include process forking and communication through shared memory.

---

8 Lawrence Snyder, "The XYZ Abstraction Levels of Poker-like Languages," in David Gelernter, Alexandru Nicolau, and David Padua, eds., *Languages and Compilers for Parallel Computing*, Cambridge: MIT Press, 1990, 470-489.

9 Mani Chandy and Steven Taylor, *Introduction to Parallel Programming*, Jones and Bartlett, 1991.

Prostor is a subset of MAYaK for statically assembling parallel modules into a parallel computation. The assemblage, called a macropipeline scheme, is described by a graph where the vertices are processes and the edges are communication channels.

MMP is the highest and most complete level, "enclosing Prostor and YaDRO." Accordingly, it is taken as synonymous with MAYaK.

The model of computation is a distributed memory, single program, multiple data (SPMD) model. Since complex problems cannot usually be solved by a single macropipeline scheme, several schemes must be invoked in succession. MPP code controls these operations and the communication between invocations is through arrays.

MAYaK has reportedly been used at the Institute to construct a parallel program for a 15,000-line industrial application. Much of the original source code, surrounded by MAYaK prolog and epilog text, could be used in building the components. The resulting program, run on the ES-2701 with 41 arithmetic processors, produced a performance improvement factor of 33 compared with the ES-1066.[10] This is a more meaningful number than those reported in the vectorizers section (III.C), and it represents a successful application of parallelism.

PARUS, Russian for parallel asynchronous recursively controllable systems, was developed at Kiev State University (Anisimov et al., 1990). Rather than being a complete language, PARUS is a set of software tools for extending base algorithmic languages such as Pascal or C together with support software. Two specificational levels are recognized. The AM level is for algorithmic modules, or processes. The CS level is for the controlling spaces, a graph formulation for resource distribution. The model of computation is multiple instruction, multiple data (MIMD). Performance data have not been noted.

---

10 Wolcott and Goodman, op. cit.

3. Functional Languages

The popularity of functional languages, which is extreme in the West, has also been substantial in the Soviet Union. Though LISP is the first widely used functional language, John Backus' FP is the language usually thought of as the archetypal functional language. Functional languages provide parallelism implicitly, since the functions that make up a program can have their parameters evaluated concurrently. Soviet experience executing functional languages in parallel has been limited. A sampling of systems includes the FPL functional programming language, the SFP stream-based functional language and two functional languages for the ES-2704, RYaD and RL.

FPL is a functional programming language around which considerable theory and a number of tools have been developed, including the FAST functional programming support system (Drobushevich and Zubovich, 1987). The functional aspects of FPL are similar to FP, but considerable attention has been paid to data structuring or "schematizing." A language called SR/TRAN is available for performing this structuring (Drobushevich et al., 1984). Within the FAST system, tools and methodologies have been developed to verify FPL programs. The AD tool, which analyzes, schematizes, linearizes, factorizes, interprets, and executes an FPL program, automates the verification process. FASE is the programming development technology with emphasis on verification that the developers claim is accessible to programmers who are not professionals (Drobushevich and Zubovich, 1990).

The following quantitative results, comparing FPL with FP, have been produced using student programmers. FPL programs are seven to eight times more "reliable" than PL/1 programs: 36 programs were written with FPL, and only two did not produce the expected results, whereas of 36 FP programs seven did not produce the expected results even with Backus style verification (Drobushevich and Zubovich, 1990).

Though these results are not directly concerned with parallel execution of the functional programs, the system is significant because of the extensive support software environment. Of course, the FPL programs can execute in parallel, given suitable hardware and compiler technology. In magnitude and goals, FAST is compara-

ble to Western efforts. The AD tool is particularly interesting since verification tools have not been the focus of much Western research in recent years.

SFP, developed in the Siberian Branch of the USSR Academy of Sciences, is a stream-oriented pure functional language for systolic and wavefront processing arrays (Trishina, 1990). The language is essentially FP semantically extended with streams. A novel feature is a geometric interpretation that can be assigned to the functions, wherein each has an associated picture with arrows indicating information flow; this characteristic is not yet fully developed. Though the goal of SFP is to facilitate rapid prototyping of systolic and wavefront algorithms, it could be a suitable language for the reconfigurable parallel computers common among Soviet architectures.

RYaD is a functional programming language designed at the Information Sciences and Automation Institute, Leningrad (now St. Petersburg) for the ES-2704.[11] The program is structured as a network with data driven control mechanisms. In addition to the expected language constituents of operators, relations, data, and references, there are resources that aid in placement or execution of program parts. Though a 24-processor prototype, ES-2704, has been produced, performance numbers for RYaD programs running on it are unavailable.

RL, a hybrid of LISP and the Soviet symbolic language REFAL, is a functional language originally developed at the Applied Mathematics Institute in Moscow and expanded by the Programming Systems Institute.[12] RL is presently being developed on a PC in anticipation of using the ES-2704. Scheduling of concurrent execution of program fragments is performed automatically at run-time.

---

11 Curran and Wolcott, op. cit.

12 Ibid.

4. Asynchronous Process Languages

Several languages supporting the "concurrent sequential processes (CSP)"[13] model of computation have been developed. Of these, Polyar is widely known and used for diverse applications.[14]

Developed in the Siberian Department of the USSR Academy of Sciences and used as a component of the START program, Polyar is an asynchronous, process-based language (Lel'chuk and Marchuk, 1983). The principal designer asserts that the language is founded on principles of the Kotov-Narin'yani asynchronic model (Lel'chuk, 1984). Processes are spawned dynamically and can have a hierarchical structure with subordinate processes communicating by argument/value and co-processes communicating via channels. Communication channels are typed, buffered, and operate in a data-driven manner. Synchronization is implemented by means of guarded commands. Processes are *a priori* parallel, and execution is asynchronous (Kreker and Lel'chuk, 1989).

Because of the asynchronous capabilities of the language, it is suitable for programming operating systems and defining architectures; both applications have been reported in connection with the MARS project (Kreker and Lel'chuk, 1989). However its object oriented character and the availability of libraries[15] imply the potential for general application programming. A version of Polyar, translating to C and built on Unix, was implemented in the mid-1980s and a descendant language, C+, has also been in use at the Siberian Branch of the USSR Academy of Sciences. No reference to physical parallel execution, as opposed to multiprogrammed logical execution on a sequential computer, has been noted. In addition to the two versions mentioned, two variants, Bundled Polyar and Polyar YaDRO for hardware and software system design,[15] have also been developed, suggesting that Polyar has had considerable impact. Indeed, when compared with OCCAM, the West's most widely known operational version of CSP, it would appear that Polyar has achieved a some-

---

13 C. A. R. Hoare, "Communicating Sequential Processes," *Communications of the ACM, 21,* 8 (1978),666-677.

14 The name "Polyar" has occurred in Western publications as "Polar" (Kotov, 1991).

15 Curran and Wolcott, op. cit.

what more workable base semantics and at the same time retained the inherent simplicity of CSP.

5. Fine-Grain Parallelism

Data parallel languages have been developed in the United States for single instruction, multiple data (SIMD) computers. Research on such languages has not been identified in the Soviet Union. However, the BARS language has features characteristic of data parallel languages, though it was not developed for a SIMD machine (Kotov, 1991) and does have multithreaded, asynchronous control.

Developed at the Siberian Branch of the USSR Academy of Sciences by a group headed by V. Ye. Kotov, BARS[16] is a fine-grain, shared memory, strongly typed parallel language designed for easy extensibility (Bul'yonkov et al., 1986). Expressions are the dominate language construct. Data manipulation expressions use generator operators to create structured data objects in a manner similar to APL and functional languages. As with these languages, BARS extends functional application to the structured objects by *propagation* (unary and binary) of scalar functions to the elements of the objects. Expressions are provided for defining the flow of control: a static graph describing "firing rules" for language statements is defined by an expression using operators for sequential composition, alternatives, looping, and concurrent composition. "Triggers" are also available for specifying data dependent control flow.

BARS provides a small set of primitive facilities with which more sophisticated structures can be constructed. BARS has reportedly been used as a design language, but not for application-oriented programming.[17]

E. KEY SOVIET RESEARCH PERSONNEL AND FACILITIES

Table III.1 provides several Soviet sites that have produced quality parallel programming research and Soviet scientists who have been particularly active in this area.

---

16 The name "BPL" for Basic Parallel Language has been used (Kotov, 1991).

17 Curran and Wolcott, op. cit..

Table III.1
KEY SOVIET RESEARCH PERSONNEL AND FACILITIES— PARALLEL PROGRAMMING

Applied Mathematics Institute im. M. V. Keldysh, USSR/Russian Academy of Sciences, Moscow (Russia)

Belorussian State University im. V. I. Lenin, Minsk (Belarus)

Gennadiy Antonovich Drobushevich
Konstantin Antonovich Zubovich

Computer Center, Siberian Branch, USSR/Russian Academy of Sciences, Novosibirsk (Russia)

Tat'yana I. Lel'chuk
Nikolay Nikolayevich Mirenkov

Control Problems Institute, USSR/Russian Academy of Sciences, Moscow (Russia)

Andrey Vladimirovich Babichev

Cybernetics Institute im. V. M. Glushkov, Ukrainian Academy of Sciences, Kiev (Ukraine)

Viktor Mikhaylovich Glushkov
Semyon Samuilovich Gorokhovskiy
Yuliya Vladimirovna Kapitonova
Aleksandr Adolfovich Letichevskiy

Cybernetics Problems Institute, USSR/Russian Academy of Sciences, Moscow (Russia)

Informatics Systems Institute, Siberian Branch, USSR/Russian Academy of Sciences, Novosibirsk (Russia)

V. Ye. Kotov

Table III.1
KEY SOVIET RESEARCH PERSONNEL AND FACILITIES—
PARALLEL PROGRAMMING (cont'd.)

Improvement of Professional Skill of Managers and Specialists Institute
(location not identified)

V. G. Lebedev

Kiev State University im. T. G. Shevchenko, Kiev (Ukraine)

Anatoliy Vasil'yevich Anisimov

F. PROJECTIONS FOR THE FUTURE

Soviet parallel programming languages and systems have advanced quite far given the limited availability of parallel hardware. To continue to advance, however, at least in the directions followed in the West, requires that computer scientists in the successor states of the former Soviet Union have convenient access to high-performance parallel hardware on a routine basis. Given the state of Soviet hardware described in Table II.1 in Chapter II, it is obvious that widespread access to high-performance parallel machines will not be provided by Soviet computers. Assuming that Western or Japanese computers fill the void—reportedly already happening with the availability of INMOS Transputers—there is a small group of well trained researchers in the successor states of the former Soviet Union who could put them to effective use. One can conjecture that if Western computers are used, the larger, more stable base of Western languages and support software will likely be employed rather than Soviet systems. Transporting certain of the Soviet systems to Western platforms, however, would not be difficult.

(blank)

CHAPTER III: PARALLEL PROGRAMMING
REFERENCES

Anisimov, A. V., Yu. Ye. Boreysha (Boreisha), and P. P. Kulyabko (Kuliabko), "PARUS Parallel Programming Technology", *Autom. Remote Control*, 6(1990), 166-173.

Babichev, A. V., and V. G. Lebedev, "Parallelization of Program Loops," *Programm. Comp. Soft.*, 5(1983), 51-55.

Babichev, A. V., V. G. Lebedev, V. V. Parshentsev, V. A. Pronina, and E. A. Trakhtengerts, "Construction of a Translator for a Multiprocessor Computer System," *Cybernetics*, 1(1985), 45-53.

Bul'yonkov (Bul'onkov), M. A., A. V. Bystrov, N. N. Dudorov, D. A. Kasperovich, and T. G. Churina, "The Experience of Realizing the Parallel Programming Language BARS on a Sequential Architecture Computer," in *Computer Systems and Software*, Eds. V. Ye. Kotov, Yu. L. Vishnevskiy (Vishnevskii), T. I. Lel'chuk, A. G. Marchuk, Siberian Branch, USSR Academy of Sciences, 1986, 78-88.

Drobushevich, G. A., To Tuan, and K. A. Zubovich, "Structured R/TRAN," *Programm. Comp. Soft.*, 3(1984), 84-89.

Drobushevich, G. A., and K. A. Zubovich, "Methods and Support Tools of Program Manufacturing in a Functional Programming System," *Programm. Comp. Soft.*, 2(1987), 8-22.

Drobushevich, G. A., and K. A. Zubovich, "Automatic Verification of Functional Programs," *Cybernetics* 4(1990), 30-39.

Glushkov, V. M., Yu. V. Kapitonova, and A. A. Letichevskiy (Letichevskii), "A Theory of Data Structures and Synchronous Parallel Computations," *Cybernetics*, 6(1976), 2-15.

Glushkov, V. M., Yu. V. Kapitonova, and A. A. Letichevskiy (Letichevskii), "Design Theory of Multiprocessor Computer Hardware and Software," *Cybernetics*, 6(1978), 811-822.

Glushkov, V. M., Yu. V. Kapitonova, A. A. Letichevskiy (Letichevskii), and S. P. Gorlach, "Macroconveyer Computations of Functions on Data Structures," *Cybernetics*, 4(1981), 439-449.

Gorokhovskiy (Gorokhovskii), S. S., Yu. V. Kapitonova, A. A. Letichevskiy (Letichevskii), I. N. Molchanov, and S. B. Pogrebinskiy (Pogrebinskii), "The Algorithmic Language MAYaK," *Cybernetics* 3(1984), 371-398.

Ivannikov, V. P., S. V. Nesterov, and A. P. Chernyayev (Cherniaev), "Comparative Study of PL/1 Compilers for a Vector Pipelining Computer," Preprint, Moscow: Nauchn. Sov. Kompleksn. Probl. "Kibernetika" AN SSSR, Moscow, 1986 (in Russian).

Konoshenko, M. P., and P. N. Telegin, "Paralleling Loop Nests in Computing Systems With A Small Number of Processors and A Common Memory, *Programm. Comp. Soft.*, 2(1989), 10-20.

Korneyev (Korneev), V. D., N. N. Mirenkov, and A. Sh. Nepomnyashchaya (Nepomniashchaia), "PARIS, Language of Superhigh Level,"*Cybernetics*, 1(1990), 34-36 (in Russian).

Kotov, Vadim Ye., "Concurrency + Modularity + Programmability = MARS," *Communications of the ACM*, 34, 6(1991), 32-46.

Kreker, G. M., and T. I. Lel'chuk, "Asynchronous Programming Tools for Multiprocessor Systems," *Programm. Comp. Soft.*, 1(1989), 35-46.

Lel'chuk, Tat'yana I., "Language Realization of a Parallel Asynchronic Computation Model," Cybernetics, 5(1984), 651-658.

Lel'chuk, Tat'yana I., and A. G. Marchuk, "Polyar Language—An Asynchronous Parallel Programming Language," *Programm. Comp. Soft.*, 4(1983), 59-68.

Mikhalevich, V. S., Yu. V. Kapitonova, and A. A. Letichevskiy (Letichevskii), "Macropipelining of Computations," *Cybernetics*, 3(1986), 261-270.

Mirenkov, Nikolay Nikolayevich, *Parallel'noye programmirovaniye dlya mnogomodul'nykh vychislitel'nykh sistem (Parallel Programming for Multimodular Computer Systems)*, Moscow: Radio i svyaz', 1989.

Trakhtengerts, E. A., *Programmnoye obespecheniye parallel'nykh protsessov (Software Support of Parallel Processes)*, Moscow: Nauka, 1987.

Trishina, Yelena, "Language Tools for Specification, Design and Rapid Prototyping of Systolic/Wavefront Algorithms," *Parallel Computing 89*, Ed.~D. J. Evans, G. R. Joubert, and F. J. Peters, North Holland: Elsevier Science Publishers, 1990, 473-479.

Val'kovskiy (Val'kovskii), V. A., and V. G. Lebedev, "Loop Vectorizers," *Autom. Remote Control*, 8(1989), 3-23.

Varchenko, V. S., and L. G. Natanson, "Development and Implementation of a Vector FORTRAN for M-10,", *Programm. Comp. Soft.*, 4(1986), 47-58.

Yershov (Ershov), A. P., "Integrated Approach to Current Program of Software Development," *Cybernetics, 20*, 3(1984), 315-328.

Zadykhaylo (Zadykhailo), I. B., S. D. Zelenetskiy (Zelenetskii), L. N. Platonova, et al. "Fora-ES: A FORTRAN IV Programming System for the PS-3000 Mulitprocessor Computing Complex," Preprint 17, Moscow: Applied Mathematics Inst., 1987.

ADDITIONAL SUGGESTED READING

Berestvaya (Berestvaia), S. N., A. B. Godlevskiy (Godlevskii), S. S. Gorokhovskiy (Gorokhovskii), Yu. V. Kapitonova, A. A. Letichevskiy (Letichevskii), and N. M. Mishchenko, "Implementation of MAYaK Family Languages for Macropipelined Multiprocessor Computing Systems," *Cybernetics*, 3(1989), 29-34.

Kalyayev (Kaliaev), A. V., O. B. Makarevich, I. A. Nikolayev (Nikolaev), and L. K. Babenko, "A Supercomputer With a Programmable Architecture," in *EVT Electronic Computer Technology*, v. 2, Ed. V. V. Przhiyalkovskiy (Przhiialkovskii), Moscow: Radio i svyaz', 1988, 161-171.

CHAPTER IV
PARALLEL ALGORITHMS AND APPLICATIONS

A. SUMMARY

This chapter addresses the topic of parallel algorithms research, is general in coverage, but focuses on numerical algorithms.

An earlier FASAC report found that, relative to other technical areas, Soviet theoretical work in some areas of applied mathematics has been first-rate and perhaps superior to that in the West.[1] Soviet researchers have placed greater emphasis on analytic modeling and error analysis than their Western counterparts. In general, what we have found in this review and assessment confirms the earlier report. These general statements carry over to the area of parallel algorithms development. That is, Soviet researchers have been good at development of the theoretical work leading up to the actual implementation, but have failed to carry the ideas forward to workable implementations. Examples of a number of approaches developed in the Soviet Union that have had a significant impact on scientific computation include multigrid and domain decomposition. These techniques are mathematical in nature and have a firm theoretical foundation. However, the lack of suitable computing hardware to develop and implement these ideas fully has retarded Soviet progress. In contrast, in the West, these ideas have evolved and been fully adapted for use on parallel computers with great success. Thus, with the apparent nonexistence of high-performance computers like those in the United States and Japan, Soviet applied mathematicians and computer scientists investigating parallel algorithms have been at a substantial disadvantage compared to their Western counterparts.

Similarly, the noticeable absence of parallel computing hardware in the Soviet Union impeded development of parallel algorithms. This situation may change with the availability of transputers from Inmos, as well as processors from computer companies such as Intel, MIPS, and SUN. In fact, this past year witnessed a surge of publications and research performed as the result of access to transputer-based

---

1 J. Bengston, R. R. Cronin, and R. B. Davidson, *The Soviet Applied Information Sciences in a Time of Change*, Foreign Applied Sciences Assessment Center (FASAC), Science Applications International Corporation, McLean, Virginia, July 1991.

computers. However, there are currently very few existing parallel applications in the successor states. This is consistent with the absence of parallel hardware and algorithms, but, again, this situation could easily change when such devices are made available.

Assessment of the published research literature in parallel algorithms suggests that Soviet researchers have been pursuing lines of investigation similar to those undertaken in the United States and Europe. These areas include core algorithms such as linear algebra and fast Fourier Transforms (FFTs), algorithms for systolic architectures, signal processing, computational fluid dynamics, and boundary value problems. With a few exceptions, however, Soviet accomplishments lagged behind those of the United States and Europe, typically by five to 10 years. In the areas of domain decomposition, and to some extent in linear algebra, Soviet research has been much closer to parity with work in the West.

B. INTRODUCTION

Computer modeling of a large variety of physical phenomena has reached the point where reality can be simulated with a useful degree of realism. Models of the atmosphere, combustion, mechanical systems, aerospace and automotive vehicles, factory operations, and economic systems can be sufficiently detailed to enable practitioners in those areas to use the models to attempt predictions. This is making high-performance computing more and more central to economic activity. Recognition of the importance of high-performance computing in solving physical problems is not limited to the West (Burtsev, 1990; Torchigin, 1990).

Basic research on parallel algorithms falls into two general categories. In the first category, existing sequential algorithms can be implemented on a parallel computer with little or no extra effort or adaption. In the second, the research focuses on creating an algorithm suitable for a parallel machine. This same trend is evident in the work performed by Soviet scientists.

The algorithms that adapt well on a parallel computer are derived from a mathematical analysis that leads naturally to parallelism. Usually the problem space is decomposed into simpler parts, each of which can be calculated by a simple proce-

dure; a straightforward, additional step puts the approximations together to give a solution of the complete, original problem.

Often the general and simplistic assessment is made that Soviet researchers are good in theory and analysis. In some areas of applied mathematical research, the Soviet work is first-rate or even superior to that in the West. A number of approaches, such as multigrid, preconditioning, and domain decomposition, can be attributed to researchers in the Soviet Union (Bakhvalov, 1966; Fedorenko 1962, 1964; Kuznetsov, 1984). These ideas were first created and documented in the Soviet Union, but not exploited to their full potential. They were later read about or rediscovered in the West, where the ideas were investigated, understood, and fully implemented as working computer programs.

The Soviet lag in the development of high-performance computers has put computer scientists and mathematicians in the successor states of the former Soviet Union at a substantial disadvantage compared to those in the West.[2] Access to parallel and high-performance computers is commonplace in the United States, allowing researchers to quickly experiment and test out ideas. In the Soviet Union, until recently, only theoretical studies could be performed.

In the West, high-performance computing is a pervasive and powerful technology for industrial design and manufacturing, scientific research, communications, and information management. Indeed, computers are the most powerful force changing modern science.[3] In the United States, high-performance computing and computer communication networks are becoming increasingly important to scientific advancement, economic competition, and national security. The technology is reaching the point of having a transforming effect on society, in industry, and in educational institutions.

To some extent, European countries realize the effects that high-performance computing can have overall and the potential problem in competitiveness with the

---

2 Ibid.

3 *The Federal High Performance Computing Program,* Washington, DC: Executive Office of the President, Office of Science and Technology Policy, 1989.

US and Japan. In fact, a recently commissioned study[4] by the European Community recommends that some $1.5 billion be spent on improving the computing industry to establish a competitive European high-performance computing industry. Nothing approaching these efforts has appeared in the Soviet Union.

Soviet universities, however, have enjoyed a strong theoretical base in mathematics, which is a critical requirement for parallel algorithm development. With the availability of low-cost processors and memory, the successor states of the former Soviet Union can be expected to make greater use of distributed computing. Software support can come from a number of outside sources, that is, the United States and Europe. There have been many Soviet research projects on heterogeneous processing. This will have the effect, over the next five years, of accelerating the development of parallel algorithms and applications development in the successor states.

In Europe, there is a strong emphasis on work being done on transputers. The transputer is a computer processor manufactured by the British company Inmos that has built-in communication links allowing processors to be put together in a straightforward fashion. The current processor is the Inmos T800 (20 MHz), which runs the Linpack Benchmark in 0.4 MFLOPS, which is equivalent to the Sun 3 processor. Results based on the use of transputers are beginning to appear in research reports from Soviet scientists. This trend will accelerate as transputers and low-cost processors appear in the successor states.

C. PARALLEL ALGORITHMS

In the West since the mid-1980s, there has been tremendous activity in the area of parallel algorithms and, to a lesser extent, parallel applications research. The number of new journals with the theme of parallel computing and an emphasis on algorithms and applications has exceeded 10 in this short time. Furthermore, there has been a sizable effort to increase the funding level for computational science and massively parallel computation. No such effort has been seen in the Soviet Union. In fact, Soviet scientists published very little in the traditional journals.

---

4 *Report of the EEC Working Group on High-Performance Computing,* Geneva: Commission of the European Communities, 1990.

Some algorithms have been adapted to run on the PS-2000, a single instruction, multiple data (SIMD) computer with 64 processors. Baklanova et al. (1988) discuss algorithms to perform digital signal filtering where the signals are to be processed in "real-time." These algorithms look rather obvious and straightforward. The only comparison given is between the PS-2000 and the ES-1061; the PS-2000 is eight times faster at performing a certain reconstructed image.

Trishina (1990) describes a set of language tools to help in the development of systolic algorithms.

1. Special Purpose Hardware for Applications

A group at the Theoretical Physics Institute im. L. D. Landau has built a Monte Carlo processor capable of performing more than 4 million elementary Monte Carlo steps per second (Talapov et al., 1990). The ideas used by the Soviet researchers have been similar to those of other projects designed in the West and dating back to 1983.[5] The Soviets have claimed that their processor is much simpler and cheaper than the original designed in 1983. The special-purpose processor is controlled by an IBM PC/AT compatible. There is no information on the hardware used to perform these computations. The Soviets have claimed that the speed is comparable to that of a supercomputer, but the cost is only a fraction of an ordinary personal computer. Similar projects have been started in the United States during the past 10 years and have produced similar results.

2. Multigrid

The multigrid method (Bakhvalov, 1966) is an example of a very powerful mathematical approach conceived by a Soviet researcher; however, the ideas could not be developed fully. One reason for this inability to develop the ideas was the lack of suitable computational tools.

Multigrid methods were originally applied to simple boundary value problems that arise in many physical applications. This approach has become a very popular

---

5 A. Hoogland, J. Spaa, B. Selman, and A. Compagner, *J. Comput. Phys.*, 51(1983), 250.

tool, in recent years, for the solution of elliptic equations. In particular, well-designed multigrid methods can yield solutions correct to the truncation error in $O(n)$ operations, where n is the number of unknowns. The three main steps in a multigrid computation are (i) a smoothing operation, (ii) the restriction to a coarser grid, and (iii) the interpolation back to a finer grid. Each of these operations is local, affecting only grid points that are neighbors, and consequently each operation is a candidate for parallelizing.

The original ideas have been extended to what are more appropriately called multilevel methods. Purely algebraic problems (for example, network problems and structural problems) have led to the development of algebraic multigrid. Identifying the notion of a grid with an array of pixels had led to multigrid applications in image processing. Multilevel methods have found new applications in control theory, combinatorial optimization (the traveling salesman problem), statistical mechanics (the Ising model), and quantum electrodynamics. The list of problems amenable to these multilevel methods is long and growing.[6] It is interesting to note that after the ideas have been developed for parallel processing, they are finding their way back into the Soviet textbooks (Marchuk, 1989).

3. September 1991—Novosibirsk Meeting

In September 1991, there was an international conference on parallel computing held in Novosibirsk. The title of the meeting was "Parallel Computing Technologies." Forty-eight papers were presented, and 30 presentations were made by researchers from the Soviet Union. Additional talks were presented by researchers from a number of other countries including Germany, Ireland, Scotland, France, Poland, China, Italy, Switzerland, and Yugoslavia.

A number of papers focused on the design of basic algorithms developed for systolic arrays such as eigenvalue computations (Likhoded et al., 1991) and solving systems of linear equations (Kosyanchuk and Yakush, 1991). These approaches are similar to those developed in the United States during the past seven years and pre-

---

6 S. McCormick, *Multigrid Methods*, Vol. 3 of the SIAM Frontiers Series, Philadelphia: SIAM, 1987.

sent nothing new. There was no mention of actual implementation or hardware to support these algorithms.

Yelizarova et al. (1991) described the adaptation of a computational algorithm onto a transputer based computer. This paper is interesting in that the authors actually develop software, implement the application on a parallel machine, and supply results. A method is implemented to the solution of Navier-Stokes equation. Navier-Stokes equation is important in three-dimensional flows in many applications, such as aircraft design. The paper discusses development of a software package called KIPARIS, as well as some computational results on a parallel computer. The authors have implemented their algorithm on a system called Hathi-2 located in Abo-Akademy in Finland. The Hathi-2 has 100 T800 transputers each with 1.25 megabytes (Mb) of memory. The Hathi-2 system has a crossbar switch making it possible to configure the machine into different topologies, such as a grid, a tree, or a torus. The other system, called the APS-48, is located in the Informatics and Computer Technology Center, in Sofia. This system is based on 32 T800 transputers, each with 2 Mb of memory. The examples described are for the flow near the step and unsteady flow in the boundary layer-shockwave interaction. The results are given as graphs showing the isobars for the shockwave, the Mach number is 2 and 2.52 and Reynolds number 1,000 and 5,000, respectively.

Kostomarov et al. (1991) make reference to the solution of magnetohydrodynamics (MHD) problems and Monte Carlo code for plasma kinetics problems. These problems come from plasma physics and represent computational-intensive problems. They describe an approach based on a hybrid model, but give very little detail of the actual methods used. The paper goes on to describe the parallel computer, ISOT-1014, which is based on four ES-2706 array processors similar to the AP-120 produced a number of years ago by Floating Point Systems Inc. The processors are fairly low in computational power by today's standards, (the Linpack performance numbers[7] are about 2 MFLOPS for the processor). The Soviet researchers claim to have a Fortran program that runs on the four-processor machine, but state that there

---

7 Jack Dongarra, *Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment*, University of Tennessee Technical Report, Computer Science Department, CS—89—85, Knoxville, 1991.

are great difficulties in programming because of the high performance and limited RAM of the array processor and low speed of the host-array processor interchange.

Overall, the meeting showed no unifying focus. It is clear from the papers presented that the work being conducted is at an early stage and that much of the work was at a primitive level with little in the way of actual implementation. The Soviet scientists appear well aware of the parallel processors available in the West (Babayan, 1991).

4. June 1990—Moscow Meeting

On 26-28 June 1990, the British-Soviet Symposium on Transputer Systems in Moscow presented a number of research projects using the transputer.[8] Overall, the work and the researchers appeared to have a good understanding of the computational models, and issues such as performance were addressed. There were many papers relating to the discussion of specific features of implementation of parallel numerical methods of linear algebra, the basic building block for the solution of many complicated problems.

A number of papers presented a more theoretical treatment of the problem. Bandman (1991) presents a space-time trade-off for cellular computations. An analysis is presented for certain computations such as iterative systems and systolic algorithms where the number of computational elements are reduced at the expense of the running time. The analysis is performed at a mathematical level and no implementation is discussed (Monakhov, 1991; Rvachev et al., 1991).

Berzigiyarov et al. (1991) describe an approach to solve ordinary differential equations (ODEs) based on a hierarchical parallelism using a collection of transputers in Prolog. The Prolog system is a front for the user interface, and the actual program is based on Occam, the native language of the transputer. A master-slave model is used in the algorithm in the ODE solver. This paper is interesting in that it puts together a number of software layers, such as Prolog and Occam, to solve the problem. The mathematical problem solution itself is not very innovative.

---

8 *SERC/DTI Transputer Initiative, Mailshot*, ed. Terry Mawby, Rutherford Appleton Laboratory, Feb 1991.

A simulator for the Inmos T414 transputer was developed by Bakhteyarov et al. (1991). This simulator runs on an IBM PC XT/AT and allows for simulations of the memory transfers and instruction set. The Soviet researchers plan to use the simulator to aid in debugging programs and also as a teaching aid in understanding the transputer. No information is provided on the speed of the simulation.

A. I. Galishkin (1991a-b) presented two papers on neural computing. One (1991a) provides an introduction to neural computing and a description of neural computers as an architectural class of computer. The other (1991b) gives a general description of a neuro computer based on a network of IMS A100 single processors and a T800 processor with a PC front-end. (The IMS A100 is a 16-bit, signal processor.) Some performance results are stated for a 16-node system doing multiplication of two vectors of size 1024.

5. PS-2000 at the Applied Physics Institute in Novosibirsk

In September 1990, a group from Sandia National Laboratories and Lawrence Livermore National Laboratory in the United States visited the Applied Physics Institute in Novosibirsk. The group was composed of Jim Asay from Sandia and Walt Herrmann and Bill Nellis from Livermore. The Applied Physics Institute in Novosibirsk was responsible for the development of advanced computing capabilities in the former Soviet Union, and performed all the major computer support for the Chemical Physics Institute at Chernogolovka, which was one of the major weapons laboratories in the Soviet Union.

One of the main computers at the Applied Physics Institute is a PS-2000 parallel computer with 128 processors in an SIMD configuration. This computer is used to perform high-velocity impact studies. During the visit, the US group was asked to define a problem to be run on the PS-2000 computer. Jim Asay set up a two-dimensional hypervelocity impact problem, which involved the impact of an aluminum pellet into a double-walled shield followed by an aluminum target. This would be a typical problem for a satellite encountering a high-speed object. The problem stresses a number of features—numerical, as well as the sensitivity of the underlying model. Jim Asay described the problem on paper, and, through a user-friendly interface, the description was given to the computer. The calculation was performed on a

coarse grid and was solved in about 12 minutes on the PS-2000. The output of the computation was a color graph showing the results.

When the US scientists returned to the United States, they reproduced the results on a Cray Y-MP using one processor. They tried two different approaches and measured the execution time. The time to solution was four minutes in the first case and 14 minutes in the second case. The run that took 14 minutes used a method that was not vectorized. A more up-to-date version would give results on the order of five to seven minutes.

The US scientists were impressed with the results from the PS-2000, in that it illustrated a degree of sophistication they had not expected. They estimated that the Soviets were two years behind the best capability in the United States.

The Soviet scientists stated that they would have three-dimensional capability in a year—that was in 1990. At a recent meeting in Japan, private conversations with Soviet researchers disclosed that they are now doing 3-D calculations, and the PS-2000 computer has been updated to have 256 processors.

D. KEY SOVIET RESEARCH PERSONNEL AND FACILITIES

A number of Soviet researchers are doing state-of-the-art work in parallel algorithms; for example, Yu. A. Kuznetsov (Scientific Research Computer Center, USSR/Russian Academy of Sciences,[9] Moscow), who does domain decomposition for problems like electromagnetics (Kuznetsov, 1984). He is well respected and on an equal footing with many US and European researchers. His algorithms are naturally parallel, but he concentrates on proving convergence theorems, and his implementations have been (necessarily) on small scalar machines. Others are V. I. Agoshkov (Moscow), A. M. Matzokin (Novosibirsk), and Sergey Nepomnyashchikh (Novosibirsk), who have made contributions to the area of domain decomposition.

A group of researchers at the Computer Center of the Siberian Branch of the USSR/Russian Academy of Sciences, in Novosibirsk, are actively working on paral-

9 At the end of 1991, the USSR Academy of Sciences changed to the Russian Academy of Sciences, the name it used before June 1925.

lel computing on a transputer-based computer. Recently, they posted a number of projects being worked on and solicited contacts and collaborations with Western scientists. This came through a publication of the Scientific and Engineering Research Council in the United Kingdom. Seven Soviet projects were listed:

- Parallel Program Synthesis—V. E. Malyshkin
- Software Tools for Fine-Grain Parallel Programming and Visualization—Ye. Pogudin
- Design and Analysis of Systolic/Wavefront Algorithms and Structures for VLSI-Implementation—S. G. Sedukhin
- Algorithms for Structure Synthesis and Subsystems Placement for Parallel Multitransputer Systems—O. G. Monakhov
- System of Parallel Program Mapping on Parallel Architectures—O. G. Monakhov
- Debugging System for Parallel Programs—S. N. Simonov
- Algorithms and Software for Modeling Electrophysical and Fluid Dynamics Problems—S. A. Sander

A brief, half-page description of each was provided. The topics cover a wide range of interests. From the information provided it was hard to tell the depth of each effort. Many projects mentioned the use of transputers.

A respected Soviet mathematician is G. I. Marchuk, former president of the USSR Academy of Sciences and a member of the Academy's Department of Computational Mathematics, Moscow. He has written extensively on numerical methods of solving problems in mathematical physics and has been at the forefront of the use of domain decomposition methods for problem solution. These approaches have direct application for many problems whose solution can be expressed in terms of partial differential equations that can be formulated and solved in parallel (Marchuk, 1988, 1989).

Another noteworthy researcher is V. V. Voyevodin (1991), who does parallel algorithms research for linear algebra, but seems to be doing rather obvious things involving breadth-first traversal of the graph of operations describing the algorithm.

Scientific Research Institute "Kvant" in Moscow, where there are a number of ongoing projects involved with transputers, seems to be home to a bed of activities.

Apart from the work noted above, there is no exciting or far advanced parallel algorithm work being done in the former Soviet Union.

Table IV.1 outlines the key Soviet research personnel and facilities discussed above.

E. PROJECTIONS FOR THE FUTURE

Parallel algorithms research in the former Soviet Union has not had the depth of similar research in the West.

It is expected that the level of research in the successor states of the former Soviet Union will grow and expand to include:

- increased access for the scientific and engineering research community through high-bandwidth networks;

- increased research in computational mathematics, software, and algorithms to make effective use of high-performance computer systems; and

- training of personnel in the scientific and engineering computing.

Today's high-performance computers are a major departure from traditional sequential machines. Improvements come from machines using perhaps several thousands of processors. In order to exploit parallel processors entirely, new algorithms must be conceived. Research in languages, algorithms, and numerical analysis will be crucial in learning to exploit these new architectures fully. The contributions of numerical analysis, computational mathematics, and algorithm design to the practice of large-scale computing is as important as the development of a new generation of machines.

There is no doubt that among scientists in the successor states of the former Soviet Union there are highly competent programmers; however, they have limited resources at their disposal.

Table IV.1
KEY SOVIET RESEARCH PERSONNEL AND FACILITIES—PARALLEL ALGORITHMS AND APPLICATIONS

Computational Mathematics Department,
USSR/Russian Academy of Sciences, Moscow (Russia)

Guriy Ivanovich Marchuk
V. V. Voyevodin

Computer Center, Siberian Branch,
USSR/Russian Academy of Sciences, Novosibirsk (Russia)

V. E. Malyshkin
O. G. Monakhov
Ye. V. Pogudin
S. A. Sander
Stanislav Georgiyevich Sedukhin
S. N. Simonov

Scientific Research Computer Center, Computational Mathematics Department,
USSR/Russian Academy of Sciences, Moscow (Russia)

V. I. Agoshkov
Yuriy Aleksandrovich Kuznetsov

Scientific Research Center/Institute "Kvant," Moscow (Russia)

Novosibirsk (Russia)
(facility not identified)

A. M. Matzokin
Sergey Nepomnyashchikh

F. SOME THINGS TO WATCH

It is fairly easy to put together low cost commodity processors to form a parallel computer. High-performance computers based on the use of commodity processors are available from a number of suppliers, such as the transputer from Inmos and i860 from Intel. These suppliers are really "packagers" in some sense. They put together a parallel system along with I/O and a network for the processors to communicate. These systems, when used as a parallel computer, can compete with traditional vector supercomputers on certain problems.

A number of companies in the West have done just that:

United States	Europe	Japan
Intel Scientific Computers (Intel i860 based)	Meiko (Intel i860 based)	Fujitsu AP-1000 (Sparc based)
Sequent, Symmetry (Intel 486 based)	Parsys (Inmos T800 transputer)	
	Parsytec (Inmos T800 transputer)	

It is entirely possible that researchers in the successor states of the former Soviet Union could follow a similar development path, if capital investments were to be made.

CHAPTER IV: PARALLEL ALGORITHMS AND APPLICATIONS
REFERENCES

Babayan (Babaian, Babajan), Boris A., "Methods of Parallel Information Processing in the Architecture of Soviet High-Performance Computers," *Proc. Int'l. Conf. Parallel Computing Technologies, 7-11 Sept 1991*, ed. N. N. Mirenkov, World Scientific, 1991.

Bakhteyarov (Bakhteiarov), S., Ye. Dudnikov, M. Yevseyev (Evseev), and A. Semyon (Semion), "A Transputer Simulator," *British-Soviet Symp. on Transputer Systems, 26-28 Jun 1990, SERC/DTI Transputer Initiative, Mailshot*, Ed. Terry Mawby, Rutherford Appleton Laboratory, Feb 1991.

Bakhvalov, N. S., "On the Convergence of a Relaxation Method and Natural Constraints on the Elliptic Operator," *USSR Comput. Math. Math Phys.*, 6(1966), 101-135.

Baklanova, O. Ye., M. V. Zyuzin (Ziuzin), and A. V. Lyulyakov (Liuliakov), "Implementation of Parallel Digital-Filtering Algorithms on the PS-2000 Multiprocessor Computational Complex," *Optoelectr. Instrum. Data Process.*, 6(1988), 3-9.

Bandman, O. L., "Space-Time Transformation of Cellular Computations," *Proc. Int'l. Conf. Parallel Computing Technologies, 7-11 Sept 1991*, Ed. N. N. Mirenkov, World Scientific, 1991.

Berzigiyarov (Berzigiarov, Berzigijarov), P., Yu. Boglayev (Boglaev), A. Karpenko, and S. Zybin, "Block Representation of Computing Algorithms Adapting with Architecture of Transputer Systems," *British-Soviet Symp. on Transputer Systems, 26-28 Jun 1990, SERC/DTI Transputer Initiative, Mailshot*, Ed. Terry Mawby, Rutherford Appleton Laboratory, Feb 1991.

Burtsev, V. S., "Trends in Development of Supercomputers," *Computers with Nontraditional Architectures: Supercomputers (Vychislitel'nyye mashiny s netraditsionnoy arkhitekturoy—Super-VM: Sbornik nauchnykh trudov)*, Ed. V S. Burtsev, Moscow: Nauka,1990, 2-23.

Fedorenko, R. P., "A Relaxation Method for Solving Elliptic Difference Equations", *USSR Comput. Math. Math Phys.*, *1*, 5(1962).

Fedorenko, R. P., "The Speed of Convergence of One Iterative Process," *USSR Comput. Math. Math Phys.*, *4*, 3(1964).

Galishkin, A. I., "On Architecture of Neural Computers," *British-Soviet Symp. on Transputer Systems, 26-28 Jun 1990, SERC/DTI Transputer Initiative, Mailshot*, ed. Terry Mawby, Rutherford Appleton Laboratory, Feb 1991a.

Galishkin, A. I., "Neurocomputers Based on Transputers and Signal Processors," *British-Soviet Symposium on Transputer Systems, 26-28 Jun 1990, SERC/DTI Transputer Initiative, Mailshot*, Ed. Terry Mawby, Rutherford Appleton Laboratory, Feb 1991b.

Kostomarov, D. P., A. V. Pedorenko, and A. M. Popov, "Construction of Parallel Programs for Multiprocessors in Computational Physics Problems," *Proc. Int'l. Conf. Parallel Computing Technologies, 7-11 Sept 1991*, ed. N. N. Mirenkov, World Scientific, 1991.

Kosyanchuk (Kosianchuk), V. V., and B. P. Yakush (Iakush), "The Design of a Linear Systolic Array for the Gauss-Jordan Algorithm with Partial Pivoting," *Proc. Int'l. Conf. Parallel Computing Technologies, Novosibirsk, 7-11 Sept 1991*, Ed. N. N. Mirenkov, World Scientific, 1991.

Kuznetsov, Yu. A., "Matrix Computational Processes in Subspaces," *Computer Methods in Applied Sciences and Engineering, VI*, Ed. R. Glowinski and J. L. Lions, North Holland: Elsevier Science Publishers, 1984, 15-31.

Likhoded, N. A., P. I. Sobolevskiy (Sobolevskii), and A. A. Tiountchik, "Design of Systolic Arrays for Iterative Algorithms: Eigenvalue Computations," *Proc. Int'l. Conf. Parallel Computing Technologies, Novosibirsk, 7-11 Sept 1991*, Ed. N. N. Mirenkov, World Scientific, 1991.

Marchuk, G. I., *Decomposition Methods (Metody rasshchepleniya)*, Moscow: Nauka, 1988.

Marchuk, G. I., *Methods of Computer Mathematics: A Textbook (Metody vychislitel'noy matematiki)*, Moscow: Nauka, 1989.

Monakhov, O. G., "Parallel Algorithm for Mapping of Program Graphs into 'Parallel Computers'," *Proc. Int'l. Conf. Parallel Computing Technologies*, 7-11 Sept 1991, Ed. N. N. Mirenkov, World Scientific, 1991.

Rvachev, V. L., A. N. Shevchenko, A. I. Koshelenko, I. I. Levin, S. Yu. Fomin), "Software and Hardware for Simulation of Physical-Mechanical Fields," *Proc. Int'l. Conf. Parallel Computing Technologies*, 7-11 Sept 1991, Ed. N. N. Mirenkov, World Scientific, 1991.

Talapov, A. L., C. B. Andreychenko (Andreichenko), Vl. S. Dotsenko, and L. N. Shchur, "Dedicated Processor for Studying Ising Model on Random Lattice," *JETP Lett., 51*, 3(1990), 182-185.

Torchigin, V. P., "Supercomputers Based on Homogeneous Medium," *Computers with Nontraditional Architectures: Supercomputers (Vychislitel'nyye mashiny s netraditsionnoy arkhitekturoy—Super-VM: Sbornik nauchnykh trudov)*, Ed. V S. Burtsev, Moscow: Nauka,1990, 121-140.

Trishina, Yelena, "Language Tools for Specification, Design and Rapid Prototyping of Systolic/ Wavefront Algorithms," *Parallel Computing 89*, Ed. D. J. Evans, G. R. Joubert, and F. J. Peters, North Holland: Elsevier Science Publishers, 1990, 473-479.

Voyevodin (Voevodin), V. V., *Parallel Software from the Standpoint of a Mathematician*, Draft, Department of Computational Mathematics, USSR Academy of Sciences, 1991.

Yelizarova (Elizarova), T. G., A. E. Dujsekulov, and L. V. Kosarev, "Kinetical-Consistent Finite-Difference Gasdynamic Schemes and Their Solution on Multiprocessor Computers," *Proc. Int'l. Conf. Parallel Computing Technologies, 7-11 Sept 1991*, Ed. N. N. Mirenkov, World Scientific, 1991.

ADDITIONAL SUGGESTED READING

Glushkov, V. M., Yu. V. Kapitonova, and A. A. Letichevskiy (Letichevskii), "Constructing a Family of Algorithmic Languages for Programming and Design of Multiprocessor Computer Systems," *Cybernetics*, 1(1981), 1-7.

Glushkov, V. M., Yu. V. Kapitonova, A. A. Letichevskiy (Letichevskii), and S. P. Gorlach, "Macroconveyor Computations of Functions on Data Structures," *Cybernetics*, 4(1981), 439-449.

Gorlach, S. P., "Design of Macropipeline Algorithms and Programs," *Cybernetics*, 3(1984), 445-447.

Kalyayev (Kaliaev), A. V., A. I. Grenchishnikov, V. F. Guzik, I. A. Nikolayev (Nikolaev), and O. B. Satishevskiy (Satishevskii), "Multiprocessor Computer Systems with Programmable Architectures," *High-Performance Systems for Parallel Processing of Information*, Vol. 4, *Parallel Processing of Information*, Ed. V. V. Gritsyk, Kiev: Naukova dumka, 1988, 100-148.

Kucherov, A. B., and E. S. Nikolayev (Nikolaev), "Parallel Algorithms for Iteration Methods with Operator Factorization for the Solution of Elliptic Boundary-Value Problems," *Differential Equations*, *20*, 7 (1984), 902-908.

Mikhalevich, V. S., Yu. V. Kapitonova, and A. A. Letichevskiy (Letichevskii), "Macropipelining of Computations," *Cybernetics*, 3(1986), 261-269.

Mirenkov, N. N., "Hierarchical Parallel Algorithms," *Artificial Intelligence and Information-Control Systems of Robots*, Ed. I. Pander, North Holland: Elsevier Science Publishers, 1984, 265-268.

Ponomarev, V. M., V. U. Plyasnin (Pliasnin), and V. A. Torgashev, *Distributed Computing and Machines with Dynamic Architecture*, Academy of Sciences of the USSR, Leningrad Research Computer Center, 1983.

Sedukhin, S. G., "Highly Parallel Algorithms and the Architecture of a Computer System for Solving Large Matrix Problems," *Artificial Intelligence and Information-Control Systems of Robots*, Ed. I. Pander, North Holland: Elsevier Science Publishers, 1984, 319-323.

Vabishchevich (Wabishchevich), A. M., N. S. Kovalenko, M. I. Ovseyets (Ovseets) and L. I. Shevchenko, "Parallel Algorithms for Graph Topology Analysis and Their Implementation for a Vector Supercomputer," *Proc. Int'l. Conf. Parallel Computing Technologies, 7-11 Sept 1991*, Ed. N. N. Mirenkov, World Scientific, 1991.

(blank)

APPENDIX A
ABOUT THE AUTHORS

Jack J. Dongarra. Dr. Dongarra is a Distinguished Scientist, specializing in numerical algorithms in linear algebra, parallel computing, use of advanced-computer architectures, programming methodology, and tools for parallel computers, at the University of Tennessee's Computer Science Department and Oak Ridge National Laboratory's Mathematical Sciences Section. He received a BS in Mathematics from Chicago State University in 1972, a MS in Computer Science from the Illinois Institute of Technology in 1973, and a PhD in Applied Mathematics from the University of New Mexico in 1978. His current research also involves the development, testing, and documentation of high quality mathematical software. He was involved in the design and implementation of the packages EISPACK and LINPACK; and is currently involved in the design of algorithms and techniques for high performance computer architectures. Other experience includes work as a visiting scientist at IBM's T. J. Watson Research Center in 1981, consultant to Los Alamos Scientific Laboratory in 1978, research assistant with University of New Mexico in 1978, a visiting scientist at Los Alamos Scientific Laboratory in 1977, as well as Senior Scientist at Argonne National Laboratory until 1989. Dr. Dongarra is a member of the Society for Industrial and Applied Mathematics (SIAM) and the Association for Computing Machinery (ACM). He has served on the SIAM Council and the ACM SIGNUM Board of Directors. He is also an editor for *Communications of the ACM, Journal of Distributed and Parallel Computing, International Journal of Supercomputer Applications, Journal of Supercomputing, Parallel Computing,* and *Research Monographs on Parallel and Distributed Computing*. He has published numerous articles, papers, reports, and technical memoranda, and has given many presentations on his research interests.

Lawrence Snyder. Dr. Snyder is Professor of Computer Science at the University of Washington. He received a bachelor's degree in mathematics and economics from the University of Iowa, and, in 1973, received a PhD in Computer Science from Carnegie-Mellon University. He was a visiting scholar at the University of Washington in 1979-80, and joined the faculty permanently in 1983, after serving on the faculties at Yale University and Purdue University. In 1987-88, he was a visiting scholar at the Massachusetts Institute of Technology and Harvard University. Dr. Snyder's research has ranged from the design and development of a 32-bit single-chip (CMOS) microprocessor, the Quarter Horse, to proofs of the undecidability of properties of programs. He created the Configurable Highly Parallel (CHiP) architecture, the Poker Parallel Programming Environment, and is co-inventor of Chaotic Routing. He is a co-developer of the Take/Grant Security Model and co-creator of several new algorithms and data structures. Following the completion of the Blue CHiP Project, he is now Principal Investigator for the Orca Project and Chief Scientist of NWLIS. Dr. Snyder is an associate editor of the *Journal of Computer and Systems Sciences* and parallel systems area editor of the *Journal of the ACM*. he served on the National Science Foundation Advisory Committee of the Division of Computer Research and dozens of other national advisory committees on future research directions.

Peter Wolcott. Mr. Wolcott is a PhD candidate in Management Information Systems at the University of Arizona. He received a BA (magna cum laude) in Computer Science and Russian from Dartmouth College in 1984. His research interests include software and high-performance computing systems in the Soviet Union. He is a member of Phi Beta Kappa, the IEEE Computer Society, and the ACM.

(blank)

APPENDIX B
GLOSSARY OF ABBREVIATIONS AND ACRONYMS

ACM	Association for Computing Machinery
ALU	arithmetic and logic unit
AM	algorithmic module
AN SSSR	*Akademiya nauk SSSR* (Soviet Academy of Sciences)
AS USSR	USSR Academy of Sciences
BESM	Russian acronym for "Large Electronic Calculating Machine" *(bystro-deystvuyushchaya elektronno-schetnaya mashina)*
CAD	computer-assisted design
CFD	computational fluid dynamics
CHiPs	configurable, highly parallel (computer)
CISC	Complex Instruction Set Computer
CMOS	complementary metal-oxide semiconductor
CS	controlling space
CSP	concurrent sequential process
DAN	dynamic automata network
ECL	emitter-coupled logic
ES	Unified System *(yedinaya sistema)*
EVM	Russian acronym for "computer" (electronic computing machine—*elektronnaya vychislitel'naya mashina)*
FFT	fast Fourier transform
FPL	functional programming language
FPS	Floating Point Systems
GFLOPS	billion floating point operations per second
GIPS	billion instructions per second
GKNT	State Committee on Science and Technology *(Obshchesoyuznyy gosudarstvennyy komitet SSSR po nauke i tekhnike)*

IEEE	Institute of Electrical and Electronics Engineers
ILP	instruction-level parallel
I/O	input/output
IPM	Applied Mathematics Institute
ITMVT	Precision Mechanics and Computer Technology Institute, Moscow
K	one thousand
kb	kilobyte
kbit	kilobit
LSI	large-scale integration
MAYaK	Russian acronym for "Macropipilined Language"
Mb	megabyte
MCS PA	multiprocessor computing system with programmable architecture
MDA	dynamic architecture machine
MESM	Russian acronym for "Small Electronic Calculating Machine" *(malaya elektronno-schetnaya mashina)*
MFLOP	million floating point operations per second
MHz	megahertz
MIMD	multiple instruction/multiple data
MIPS	million instructions per second
MKP	Modular Pipeline Processor
MMP	mnemonic for multimodular programming
Mword	million words
NIIMVS	Multiprocessor Computer Systems Scientific Research Institute, Taganrog
NITsEVT	Electronic Computing Technology Scientific Research Center, Moscow
NPO	scientific production association *(nauchno-proizvodstvennoye ob"edineniye)*
ns	nanosecond
ODE	ordinary differential equation

PARUS	Russian acronym for "Parallel Asynchronous Recursively Controllable Systems"
PC	personal computer
PE	processing element
PS	reconfigurable system
ps	picosecond
PVK	problem-oriented computing system
RAM	random access memory
R&D	research and development
RISC	reduced instruction set computer
SIAM	Society for Industrial and Applied Mathematics
SIMD	single instruction/multiple data
SISAL	Streams and Iterations in Single Assignment Language
SISD	single instruction/single data
SM	Small System *(sistema mini-EVM)*
SO	*Sibirskoye otdeleniye* (Siberian Branch of USSR Academy of Sciences)
SPMD	single program, multiple data
SRI	scientific-research institute (in Russian, NII/*nauchno-ssledovatel'skiy institut*)
STA	Soviet Transputer Association
TRAM	transputer plus associated memory
UkSSR	Ukraine (formerly, Ukrainian Soviet Socialist Republic)
VLIW	very long instruction word
VLSI	very-large-scale integration

(blank)

APPENDIX C
SOVIET JOURNALS CITED IN TEXT/REFERENCES

For readers not familiar with the Soviet technical literature, a key to the abbreviated titles of the Soviet serial literature cited in this report is provided below. The titles of the English language translations used are listed in **bold print** and the original Russian language titles are in *italics*. When a given Soviet technical journal is published in more than one commercial translation, the English title for the same Soviet source may vary with the publisher. If translations have been made privately (for example, government agency translations), the titles may also vary. Frequently, English titles are not literal translations of the original Russian. Therefore, knowledge of the Russian title of a journal may be necessary to identify reference materials.

Abbreviation	English Translation Title/*Original Russian Title*
	Applied Information Science (not translated commercially) *(Prikladnaya informatika)*
Autom. Control Comput. Sci	**Automatic Control & Computer Sciences** *(Avtomatika i vychislitel'naya tekhnika)*
Autom. Monit. Meas.	**Automatic Monitoring & Measuring** *(Avtometriya)* (in 1984, English translation title changed to/see **Optoelectronics, Instrumentation & Data Processing**)
Autom. Remote Control	**Automation & Remote Control** *(Avtomatika i telemekhanika)*
Cybernetics	**Cybernetics** *(Kibernetika)*
	Differential Equations *(Differentsial'nyye uravneniya)*
	Electronic Modeling (not translated commercially) *Elektronnoye modelirovaniye*
Instrum. Exp. Tech.	**Instruments & Experimental Techniques** *(Pribory i tekhnika eksperimenta)*
JETP Lett.	**JETP Letters** *(Pis'ma v zhurnal eksperimental'noy i teoreticheskoy fiziki)*
	Management Systems & Machines (not translated commercially) *(Upravlyayushchiye sistemy i mashiny)*
Optoelectr. Instrum. Data Process.	**Optoelectronics, Instrumentation & Data Processing** *(Avtometriya)* (prior to 1984, English translation title was/see **Automatic Monitoring & Measuring**)

Programm. Comp. Soft.	**Programming & Computer Software** *(Programmirovaniye)*
Strength Mater.	**Strength of Materials** *(Problemy prochnosti)*
USSR Comput. Math. Math Phys.	**USSR Computational Mathematics & Math Physics** *(Zhurnal vychislitel'noy matematiki i matematicheskoy fiziki)*

APPENDIX D
SOVIET RESEARCH FACILITIES CITED IN TEXT
(* full information not available)

Applied Mathematics Institute im. M. V. Keldysh, USSR/Russian Academy of Sciences, Moscow (Russia)
(Institut prikladnoy matematiki imeni M. V. Keldysha, AN SSSR—IPM)

Applied Physics Institute, Novosibirsk (Russia)
(Institut prikladnoy fiziki)

Applied Problems of Mechanics & Mathematics Institute, Ukranian Academy of Sciences, L'vov (Ukraine)
(Institut prikladnykh problem mekhaniki i matematiki, AN UkSSR—IPPMM)

Aviation Institute im. Sergo Ordzhonikidze, Moscow (Russia)
(Moskovskiy aviatsionnyy institut imeni Sergo Ordzhonikidze—MAI)

Aviation Instrument Building Institute, Leningrad/St. Petersburg (Russia)
(Leningradskiy institut aviatsionnogo priborostroyeniya—LIAP)

Belorussian State University im. V. I. Lenin, Minsk (Belarus)
(Belorusskiy gosudarstvennyy universitet imeni V. I. Lenina)

Computational Mathematics Department, USSR/Russian Academy of Sciences, Moscow (Russia)
(Otdel vychislitel'noy matematiki, AN SSSR)

Computer Center, Siberian Branch, USSR /Russian Academy of Sciences, Novosibirsk (Russia)
(Vychislitel'nyy tsentr, SO AN SSSR—VTs)

Computing Systems Scientific Research Institute, Moscow (Russia)
(Nauchno-issledovatel'skiy institut vychislitel'nykh kompleksov—NIIVK)

Control Problems Institute, USSR/Russian Academy of Sciences, Moscow (Russia)
(Institut problem upravleniya, AN SSSR—IPU)

Cybernetics Institute im. V. M. Glushkov, Ukrainian Academy of Sciences, Kiev (Ukraine)
(Institut kibernetiki imeni V. M. Glushkova, AN UkSSR)

Cybernetics Problems Institute, USSR/RussianAcademy of Sciences, Moscow (Russia)
(Institut problem kibernetiki, AN SSSR)

Electronic Computing Technology Scientific Research Center, Moscow (Russia)
(Nauchno-issledovatel'skiy tsentr elektronnoy vychislitel'noy tekhniki—NITsEVT)

High-Energy Physics Institute, Protvino (Russia)
(Institut fiziki vysokikh energiy)

"Impul's" Scientific Production Association (NPO), Control Computer Scientific Research Institute, Severodonetsk (Ukraine)*
(Nauchno-proizvodstvennoye ob"edineniye "Impul's"—NPO Impul's, Nauchno-issledovatel'skiy institut upravlyayushchikh vychislitel'nykh mashin—NIIUVM)

Improvement of Professional Skill of Managers and Specialists Institute *
(Institut povysheniya kvalifikatsii rukovodyashchikh rabotnikov i spetsialistov)

Informatics [Information Processing] and Automation Institute (Computer Center), USSR/Russian Academy of Sciences, Leningrad/St. Petersburg (Russia)
(Institut informatiki i avtomatiki/Leningradskiy vychislitel'nyy tsentr, AN SSSR—LVTs)

Informatics [Information Processing] Problems Institute, USSR/RussianAcademy of Sciences, Moscow (Russia)
(Institut problem informatiki, AN SSSR—IPIAN)

Informatics [Information Processing] Systems Institute, Siberian Branch, USSR/Russian Academy of Sciences, Novosibirsk (Russia)
(Institut sistem informatiki, SO AN SSSR)

International Center for Informatics [Information Science] and Electronics, Moscow (Russia)
(Mezhdunarodnyy tsentr po informatike i elektronike—InterEVM)

Joint Institute for Nuciear Research (JINR), Dubna (Russia)
(Ob"yedinennyy institut yadernykh issledovaniy—OIYal)

Kiev Polytechnic Institute (im. 50th Anniversary of the Great October Socialist Revolution), Kiev (Ukraine)
(Kiyevskiy politekhnicheskiy institut imeni 50-letiya velikoy oktyabrskoy sotsialisticheskoy revolyutsii—KPI)

Kiev State University im. T. G. Shevchenko, Kiev (Ukraine)
(Kiyevskiy gosudarstvennyy universitet imeni T. G. Shevchenko)

Leningrad Research Computing Center, Leningrad/St. Petersburg (Russia)
(Leningradskiy nauchno-issledovatel'skiy vychislitel'skiy tsentr)

Mathematical Machines Scientific Research Institute, Yerevan (Armenia)
(Yerevanskiy nauchno-issledovatel'skiy institut matematicheskikh mashin—YerNIIMM)

Mathematics Institute, Siberian Branch, USSR/Russian Academy of Sciences, Novosibirsk (Russia)
(Institut matematiki, SO AN SSSR)

Moscow State University im. M. V. Lomonosov, Moscow (Russia)
(Moskovskiy gosudarstvennyy universitet imeni M. V. Lomonosova—MGU)

Multiprocessor Computer Systems Scientific Research Institute, Taganrog (Russia)
(Nauchno-issledovatel'skiy institut mnogoprotsessorno-vychislitel'nykh sistem—NIIMVS)

Precision Mechanics and Computer Technology Institute im. S. A. Lebedev, USSR/Russian Academy of Sciences, Moscow (Russia)
(Institut tochnoy mekhaniki i vychislitel'noy tekhniki imeni S. A. Lebedeva, AN SSSR—ITMVT)

Programming Systems Institute, Pereslavl'-Zalesskiy (Russia)*

Radio Engineering Institute im. V. D. Kalmykov, Taganrog (Russia)
(Taganrogskiy radiotekhnicheskiy institut imeni V. D. Kalmykova—TRTI)

Scientific Research Institute/Center "Kvant" *
(Nauchno-proizvodstennoye ob"yedineniye/nauchno-issledovatel'skiy instituttsemtr Kvant)

Semiconductor Electronics Institute, Novosibirsk (Russia)
(Institut poluprovodnikovoy elektroniki)

Theoretical Physics Institute im. L. D. Landau, AS/Russian USSR, Chernogolovka (Russia)
(Institut teoreticheskoy fiziki imeni L. D. Landau, AN SSSR—ITF)

(blank)

APPENDIX E
FASAC REPORT TITLES

(* asterisk before title indicates report is classified)

(completed)

FY-82/83	*	Soviet High-Pressure Physics Research
		Soviet High-Strength Structural Materials Research
		Soviet Applied Discrete Mathematics Research
	*	Soviet Fast-Reaction Chemistry Research
FY-84		Soviet Physical Oceanography Research
		Soviet Computer Science Research
		Soviet Applied Mathematics Research: Mathematical Theory of Systems, Control, and Statistical Signal Processing
		Selected Soviet Microelectronics Research Topics
	*	Soviet Macroelectronics (Pulsed Power) Research
FY-85		FASAC Integration Report: Selected Aspects of Soviet Applied Science
		Soviet Research on Robotics and Related Research on Artificial Intelligence
		Soviet Applied Mathematics Research: Electromagnetic Scattering
	*	Soviet Low-Energy (Tunable) Lasers Research
		Soviet Heterogeneous Catalysis Research
		Soviet Science and Technology Education
		Soviet Space Science Research
		FASAC Special Report: Effects of Soviet Education Reform on the Military
		Soviet Tribology Research
		Japanese Applied Mathematics Research: Electromagnetic Scattering
		Soviet Spacecraft Engineering Research
		Soviet Exoatmospheric Neutral Particle Beam Research
		Soviet Combustion Research
		Soviet Remote Sensing Research and Technology
		Soviet Dynamic Fracture Mechanics Research
FY-86/89		Soviet Magnetic Confinement Fusion Research
		Recent Soviet Microelectronics Research on III-V Compound Semiconductors
		Soviet Ionospheric Modification Research
		Soviet High-Power Radio Frequency Research
		Free-World Microelectronic Manufacturing Equipment
		FASAC Integration Report II: Soviet Science as Viewed by Western Scientists

(completed/cont'd.)

FY-86/89		Chinese Microelectronics
		Japanese Structural Ceramics Research and Development
		System Software for Soviet Computers
		Soviet Image Pattern Recognition Research
		West European Magnetic Confinement Fusion Research
		Japanese Magnetic Confinement Fusion Research
	*	Soviet Research in Low-Observable Materials
		FASAC Special Study: Comparative Assessment of World Research Efforts on Magnetic Confinement Fusion
		FASAC Special Study: Defense Dependence on Foreign High Technology
		Soviet and East European Research Related to Molecular Electronics
		Soviet Atmospheric Acoustics Research
		Soviet Phase-Conjugation Research
		FASAC Special Study: Soviet Low Observable/Counter Low Observable Efforts: People and Places
		Soviet Oceanographic Synthetic Aperture Radar Research
		Soviet Optical Processing Research
		FASAC Integration Report III: The Soviet Applied Information Sciences in a Time of Change
		Soviet Precision Timekeeping Research and Technology
		Soviet Satellite Communications Science and Technology
		West European Nuclear Power Generation Research and Development
		FASAC Special Study: Non-US Artificial Neural Network Research (I)
	*	Radiation Cone Research in the Former Soviet Union
FY-90/91		Soviet Chemical Propellant Research and Development
		Parallel Processing Research in the Former Soviet Union

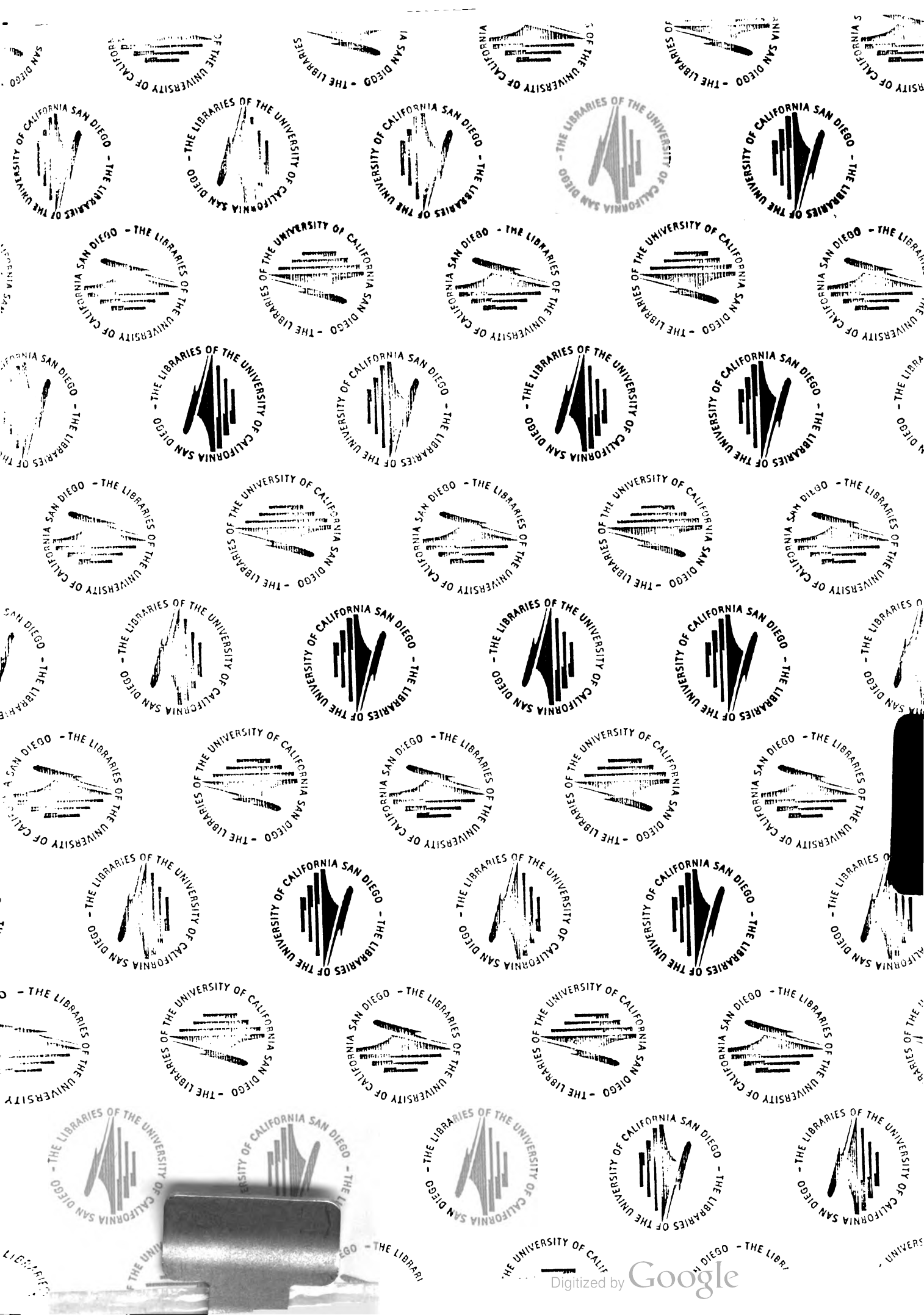
(in production)

FY-90/91		Optoelectronics Research in the Former Soviet Union
		Nonlinear Dynamics Research in the Former Soviet Union
	*	Foreign Research Relevant to Countering Stealth Vehicles
		Penetration Mechanics Research in the Former Soviet Union
		Foreign Bandpass Radome Research and Development
		FASAC Special Study: Non-US Artificial Neural Network Research (II)
		Pulsed Power Research in the Former Soviet Union
		Climate Research in the Former Soviet Union

(in production/cont'd.)

Foreign Research in and Applications of Heavy Transuranics
Foreign Research in Data Compression and Coding

(blank)


THE LIBRARIES OF THE UNIVERSITY OF CALIFORNIA SAN DIEGO
Digitized by Google

SFO DIRECTOR S.I.O. 1

SCRIPPS INSTITUTE OF OCEANOGRAPHY
A-010
LA JOLLA, CA 92093

JUN 12 1992

ATTN:
DR E A FRIEMAN, DIR